Development and Evaluation of Neural Network-Based Virtual Air Data Sensor for Estimation of Aerodynamic Angles

Politecnico di Torino - I Facoltà di Ingegneria

Doctor of Philosophy in Aerospace Engineering
XXV cycle

Development and Evaluation of Neural Network-Based Virtual Air Data Sensor for Estimation of Aerodynamic Angles

Ph.D Candidate
Angelo Lerro
Matricola: 170193

Academic Advisors
Prof. Piero Gili
Prof. Manuela Battipede
Alenia Aermacchi Advisor
Mr. Silvio Caselle

Academic Year 2012

To my Mum and Dad

This page intentionally left blank.

Acknowledgment

The present work has been performed through a close cooperation with Alenia Aermacchi S.p.A. and through the use of Confidential Information and Data, property of Alenia Aermacchi S.p.A., which remains the sole owner of all such relevant IP rights. The result of the present work shall be, therefore, property of Alenia Aermacchi S.p.A.

This page intentionally left blank.

Table of Contents

Abstract

Aerodynamic angles of flight vehicles are necessary to pilot and automatically control of aircraft. These angles are usually measured using probes that protrude from the vehicle surface out into the flow field. However, this arrangement was found to be unacceptable for modern unmanned airplanes whenever stealthiness features are required. In addition, redundant sensor arrangements, when dictated by safety regulations, were also critical because of the possible heavy impact on the airframe of small UAVs. New virtual software-based systems were therefore developed in order to find a viable solution for reducing the number of traditional hardware-based air data sensors, and they offered the benefit of simplifying air data system architectures. The aerodynamic angles were derived from inertial data and by exploiting the airspeed sourced by the Pitot-static system. The relationship between these parameters and the aerodynamic angles was a complex, non-linear function that was not easily described by means of aircraft models. The main goal of this work, which was aimed at UAV applications, was to analyze the aircraft system and develop virtual sensors by exploiting soft computing methods, such as neural prediction techniques, in order to assess the feasibility of this kind of neural system. The performance of virtual sensors were tested using real hardware in the simulation loop and to represent real-world flight conditions: wind gusts, air turbulence and internal sensor noise were simulated. A sensitivity analysis was carried out to study the performance of virtual sensors even when realistic accuracy of measured signals, processed by neural networks, and failure modes were simulated.
Finally, neural networks resulted to be suited for aerodynamic angle estimation technique: the neural networks worked properly with the available vehicle data and demonstreted to be as accurate as traditional probes.

Introduction

"What is particularly significant is that in both consumer products and industrial systems, the employment of soft computing techniques leads to systems which have high MIQ (Machine Intelligence Quotient). In large measure, it is the high MIQ of SC-based systems that accounts for the rapid growth in the number and variety of applications of soft computing"

Lotfi Zadeh - 1994

The flight control computer (FCC) can be considered the core of modern UAVs since several autopilot modes are implemented to guarantee stability, control and navigation of the aircraft, even in automatic mode. Autopilots, which have the purpose of guaranteeing automatic control, need several input parameters, such as air data, whose measurements are taken from air flow surrounding the aircraft: the angle of attack, α, and sideslip, β, which are also known as *aerodynamic angles*, are two such parameters.

Today, aerodynamic angles can be measured using vanes, which were first described in detail by Ikard [1], in 1956, for both subsonic and supersonic applications. Another way is to use differential direction probes, as was well documented by Chue [2], Pankhurst and Holder [3] and Yajnik and Gupta [4], starting back in 1952. These kinds of probes were first introduced onto the market by *Rosemount Inc.* in 1963. Most modern flow direction probes are today integrated in multifunction probes [5], i.e. probes which have the capability of sensing both aerodynamic angles and static and dynamic pressure, but the background theory is still the same. Some examples exist where the angle of sideslip of an airplane is measured, when necessary, by differentiating between two static pressures sensed on opposite sides of an aircraft, exploiting the same basic principle of multi-hole probes. These

air data sensors are connected to an air data computer (ADC) which provides the FCS with the required parameters. In all these cases, the basic flow angle measurement principles were already well known in the first half of XX century. The evolution in air data instrumentations that was taken over the last decade reflects the radical changes that have occurred in electronic measuring techniques of ADCs and transducers, from aneroid capsules to modern MEMS transducers. Nevertheless, the basic principles of air data measurement methods have essentially remained unchanged and pressure and flow measurement sensors have undergone only slight changes. The placing of air data sensors is somewhat problematic on UAVs, due to potential interference issues with opto-electronic sensors, because the best location for both kinds of sensors for thei optimal operations is on the front fuselage of the aircraft. These problems are further enhanced when a multiple installation of the same systems is requested for redundancy to comply with airworthiness regulations. Moreover, redundant ADS may also be needed for voting and monitoring capability that cross-compares the signals from different channels for detecting and isolating ADS failures at a single single sensor level, depending on the level of redundancy requested. The increasing need of modern UAVs to keep the costs and complexity of on-board systems down has encouraged the practice of substituting, whenever feasible, expensive, heavy and sometimes even voluminous hardware devices with executable software codes. Another practical example, which is referred to as *analytical redundancy* in the current literature, is the process of replacing some of the actual sensors with virtual sensors, which can be used as voters in redundant or simplex sensor systems, to detect inconsistencies of the hardware sensors and can eventually be employed to provide alternative data. More generally, analytical redundancy is identified with the functional redundancy of the system. The idea of using software algorithms to replace physical hardware redundancy was introduced as soon as digital computers started being used in the 1970's to perform redundancy management. Approaches developed to detect and isolate sensor failures were ultimately to become important parts of later control reconfiguration schemes. An example is the Sequential Probability Ratio Tests that were carried out on the F-8 Fly-by-Wire demonstrator in the late 1970's [6]. Throughout the 1970's and 1980's, many papers appeared describing various algorithms that could be used to manage redundant systems and redundant sensors. Many attractive advanced algorithmic solutions have been proposed, especially over the last two decades,

mostly related to model-based techniques, but which are capable of taking into account some robustness requirements with respect to exogenous disturbances and model uncertainties. In 2000, Napolitano et al. [7] and Oosterom and Babuska [8] independently described fault tolerant systems using soft computing techniques.
In the present work, virtual sensors will be designed with the aim of indirectly calculating aerodynamic angles exploiting soft computing techniques. The present virtual air data sensors are based on neural networks (NNs) to overcome discrepancies between the mathematical model and the real aircraft of the model-based methods, which is the main drawback of this technique, but also to define economic hardware processing systems. This kind of activity is not a novelty in the aerospace field, since several practical applications of NNs exist and, more in general, of soft computing techniques, these are used as system identification devices to estimate the angle of attack [9, 10] and sideslip [11] from data derived from other sources, without exploiting classical methods, such as vanes or modern multifunction probes. Rohloff et al. [12] and Samy and Green [13] described virtual sensors, based on neural networks, that are able to reconstruct complete suite of air data parameters, starting from multiple static pressure measurements on an aircraft fuselage, without using inertial data. Other examples of virtual sensors exist which have been developed on model based techniques, such as the one designed and patented by Wise [14], which is actually used on the Boeing X-45A aircraft. Using inertial data, in addition to an accurate aerodynamic aircraft model and a Kalman filter, the virtual sensor is able to predict the aerodynamic angles with good accuracy. Overall, in the current literature about virtual sensors for aerodynamic angle estimation, whatever the technology on which the virtual sensors are built, they all share the use of dynamic pressure actual values, which is clearly a fundamental air data that is quite complex to estimate. However, only a few examples of virtual sensors that do not need dynamic pressure exist, e.g. that patented by McCool and Haas [15], mainly dedicated to applications on helicopters. Indeed, they invented a virtual sensor able to estimate flight parameters using a reconstructed value of dynamic pressure starting from, inter alia, the exact position of the center of gravity, the engine torque on the shaft and the actual thrust in real time. Even though the results presented by McCool and Haas are good and encouraging, the parameters they use as input for virtual sensor are not always available with the required accuracy.
The novelty introduced in this research project, with respect to current literature,

is that the aerodynamic angles are estimated indirectly by means of neural systems which need inertial data from AHRS, dynamic pressure from ADS and aircraft commands from FCS as input data. It is clear that the use of virtual sensors does not preclude the use of dynamic pressure estimated by exploiting, for example, another virtual sensor. The virtual sensor solution allows one to save, or substitute, physical sensors with software-based ones and this leads to enormous benefits for the redundant systems of unmanned aircraft.

The final goal of this work is to assess the feasibility of a virtual sensor, based on neural prediction techniques, for aerodynamic angle estimation in UAV applications, while taking well defined requirements into cosiderations. The virtual sensor was mainly designed using two aircraft simulators: firstly, the well known FDC toolbox of Matlab was used to set up neural networks using an open source simulator which could be modified to satisfy any user request and which could also be run on a personal computer. The second simulator, developed by Alenia Aermacchi to train pilots and ground operators on the unmanned Sky-Y aircraft , was used to train and test neural networks for the final performance assessment, using a real flight control computer within the simulation loop.

During this research activity, several attempts had been made to represent the real-life world were done in order to assess the estimation performance of the current virtual sensors in realistic flight conditions. One of the most critical issues related to the actual neural network applications, is the gap between simulated input signals and real input signals to NNs, which have their own accuracy and noise level, and which must be taken into account to evaluate the actual performance of the virtual sensors based on neural networks. When real tests cannot be performed or there is a lack of real data, these data are replaced with noise corrupted signals, as was done by Svoboda et al. [16] and Rowley et al. [17] to understand the behaviour of the neural network when fed by actual input signals. Sensor noise was modeled to simulate errors stemming from aboard actual sensors; in particular, each input signal was characterized with its own noise level, according to available literature. Moreover, external disturbances, such as wind gusts and turbulence, were also simulated using the well known *Dryden* turbulence models.

The following chapters document the successful development of a software based on neural networks for aerodynamic angle estimation, which indirectly measure α and β for Alenia Aermacchy Sky-Y UAV. Chapter 1 starts with a general discussion of air

data measurements and ends with a review of the current state of the art of air data systems (ADSs). Chapter describes the reasons that drove this research towards the choice of neural predictive techniques for virtual sensors; the consequences of this decision are well described highlighting benefits and drawbacks of applications of virtual sensor for UAVs with reference to the state of the art of ADS technology. Chapter 3 provides a general discussion on neural network techniques, along with description of the specific techniques used during this investigation and an application to a simple test case. The details of the whole NN design process is provided in chapter 4, by exploiting the FDC Matlab toolbox. Moreover a preliminary NN performance analysis is reported and the virtual sensors were also tested when electronic noise and air turbulence were simulated. The virtual sensors were tested in chapter 5 using the Alenia Aermacchi Sky-Y UAV simulator, and in particular, once downloaded onto actual FCC, they were tested using real hardware in the simulation loop. In the last chapter a sensitivity analysis is presented in order to evaluate the importance of all input-output relationships and the robustness of virtual sensors when realistic accuracy models of measured signals, processed by neural networks, and failure modes were simulated.

Nomenclature

Symbols

a	acceleration calculated in the body reference frame
b	bias of NN
C	output signal set
C	aerodynamic coefficients
CAS	calibrated airspeed $[kts]$
C_p	pressure coefficient, $\frac{p_{s,i}-p_{s\infty}}{q_{c\infty}}$
d	desired output
e	error, $e = d - \hat{y}$
$\mathscr{E}$	error energy, $\frac{1}{2}e^2$
F	force calculated in the body reference frame
g	gravity acceleration, 9.81 m/s^2
H	height $[m]$
IAS	indicated airspeed $[kts]$
M	Mach number
n	neuron
n	number of g's
p	roll rate
p_0	total pressure $[Pa]$
p_s	static pressure $[Pa]$
q	pitch rate
q_c	dynamic pressure $[Pa]$
QFE	static pressure at ground level of runway
QNH	static pressure at sea level
r	yaw rate
R	ideal gas constant
t	time

T	temperature
T_0	total temperature $[K]$
T_s	static temperature $[K]$
TAS	true airspeed $[kts]$
u	axial component of relative velocity in the body reference system
v	lateral component of relative velocity in the body reference system
V	velocity
V_D	design dive speed
V_{NE}	never exceed speed
K_β	slope of linear approximation of β with respect to differential pressure
y	generic output
x	generic input
w	synaptic weight of NN
w	vertical component of the relative velocity in the body reference system

Acronyms

A/C	aircraft
ABS	absolute
ADC	air data computer
ADS	air data system
ADU	air data unit
AFCS	aircraft flight control system
AHRS	attitude and heading reference system
AI	artificial intelligence
ANN	artificial neural network
AOA	angle of attack
AOS	angle of sideslip
BP	back-propagation
CFD	computationsl fluid dynamics
CG	center of gravity
DIFF	differential
EXE	execution
INS	inertial navigation system
ISA	international standard atmosphere

FCC	flight control computer
FCS	flight control system
FL	fuzzy logic
FL	flight level
FS	full-scale
GA	genetic algorithm
LM	Levenberg-Marquardt
MISO	multi input single output
MSE	mean square error
MTOW	max take off weight
NED	North-East-Down
NN	neural network
NNA	neural network for α estimation
NNAAE	neural network-based aerodynamic angle estimator
NNB	neural network for β estimation
OAT	outside air temperature
PEC	pressure error correction
SAT	static air temperature
SC	soft computing
SISO	single input single output
UAS	unmanned aerial system
UAV	unmanned aerial vehicle
UCAV	unmanned combat air vehicle

Greek Ssymbols

α	angle of attack $[deg]$
β	angle of sideslip $[deg]$
β	slope of linear approximation of atmospheric temperature with altitude
δ_a	aileron deflection
δ_f	flap deflection
δ_e	elevator deflection
δ_r	rudder deflection
δ	local gradient in the BP algorithm
η	learning rate
ψ	yaw angle
ϕ	regression vector
ϕ	roll angle

θ	vector containing the free parameters of mathematical model
θ	pitch angle
ρ	air density $[N/m^2]$

Subscripts and Superscripts

$\dot{(\,)}$ time derivative

$\hat{}$	estimated value
0	total
∞	free stream condition
day	current day
exe	execution
L	left
loc	local
m	measured
max, min	maximum, minimum value
R	right
s	static
s	stall
SL	sea level
std	ISA standard
T	true value
x, y, z	longitudinal, lateral, vertical body axes

Chapter 1

Air Data System

An ADS has to provide pilots, FCSs and other on-board systems with several quantities that are derived from external air flow measurements through signals from aerodynamic and thermodynamic probes. These air data probes measure the characteristics of the air locally surrounding the aircraft and then dedicated transducers, which are integrated in the Air Data Computer (ADC) or in the sensors themselves, convert the measurement values into electrical signals. The ADC contains the correction algorithms that are necessary to calibrate, or convert, local measurements into free stream ones. The signals from transducers are calibrated and then processed to calculate all the required quantities. Since an ADS is vital for aircraft, in order to comply with safety regulations regarding airworthiness certification, sometimes two or even three are installed on some aircraft for specific safety requirements. This is particularly true for modern UAVs that have to carry out aerial work over populated areas: in this case, they could be required to comply with even more stringent airworthiness regulations than those applicable to manned aircraft (e.g. AER-P2 [18] and AER-P6 [19] in Italy).

Air data probes and sensors, which were discussed in great detail by Gracey [20] and Wuest [21] and which work solely with air flow pressure or temperature, operate direct measurements of air data parameters. These direct measurements from the air surrounding the aircraft are: the static pressure (p_s), the total pressure (p_0), the total or static temperature (T_0 or T_s respectively) and air flow angles with respect to the fixed reference axis of the aircraft body.

The static pressure is used to calculate the barometric height, H.

The total pressure, in addition to the static pressure, is used to calculate the relative velocity between the aircraft and the external air flow, which is presented to pilots as indicated, calibrated or true airspeed (IAS, CAS and TAS respectively). The IAS can be calculated from the measured dynamic pressure ($q_{\infty,m} = p_0 - p_s = {}^1\!/\!_2\rho_\infty V_\infty^2$), with reference to sea level conditions, as

$$IAS = \sqrt{\frac{2q_{c\infty,m}}{\rho_{SL}}}.$$

Correcting the dynamic pressure measurement from calibration errors (mainly due to *position* errors [22]), the IAS can be converted into calibrated air speed as

$$CAS = \sqrt{\frac{2q_{c\infty}}{\rho_{SL}}}.$$

Finally, the actual air density value can be calculated using a temperature sensor, and then the true air speed can be calculated as

$$TAS = \sqrt{\frac{2q_{c\infty}}{\rho_{\infty}}}.$$

The CAS, or IAS, are fundamental for piloting and controlling the aircraft, while the TAS is important for navigation purposes.

The Mach number calculation is obtained from the static and total pressure by using the iso-entropic relation

$$M = \sqrt{\frac{2}{\gamma - 1}\left[\frac{p_s}{p_0}^{\gamma - 1/\gamma} - 1\right]}.$$

where the effect of temperature on γ is commonly neglected for the Mach number calculation for altitudes within 36000 ft

The air flow angles are direct measurements of the local angle of attack, α_{loc}, and angle of sideslip, β_{loc}. The free stream values, α_∞ and β_∞, are then obtained using adequate calibration algorithms. Sensing the angle of sideslip is usually carried out by means of vanes as for the angle of attack α, or, through less frequently, by exploiting differential static pressure measurements from the two aircraft sides and adopting a calibration law which converts the differential pressure measurement into the angle of sideslip calculation. Owing to the x-axis symmetry of an aircraft, the aforesaid calibration law is generally presented as

$$\beta = K_\beta(p_{s,L} - p_{s,R}), \tag{1.1}$$

where K_β is usually a function of the angle of attack, the Mach number and also of the sideslip itself which takes into account the non linear aerodynamic effects of the fuselage nose that occur at high attitudes and in the presence of high lateral wind. A practical example will be given in section 4.3

A particular aspect of the angle of sideslip calculation must be addressed. This angle is no usually sensed on civil aircraft, because it is considered a trivial information for pilots to control the vehicle: a turn indicator has a more immediate effect than the numerical indication of the angle of sideslip. The angle of sideslip (and as well

Figure 1.1: An example of unmanned flying wing vehicle. The UCAV Dassault *nEUROn*

as the angle of attack) is used on some fast-dynamics aircraft, such as military ones, by the FCS for stability and augmentation purposes. The same issues are relevant for UAVs, where, due to the absence of human pilots on board, a large amount of information is needed by the FCS to fly an aircraft automatically and safely. The angle of sideslip is especially required for intrinsically stable unmanned flight vehicles during the landing and takeoff phases when there is cross-wind. The angle of sideslip is required by the flight control system for unstable UAVs, e.g. on flying wing models, because of the lack of stability on the vertical axis which is caused by their peculiar aerodynamic configuration. Several unstable unmanned models exist such Unmanned Combat Air Vehicles (UCAVs) Boeing X-45, Dassault *nEUROn*(see Fig. 1.1) and Northrop Grumman X-47.

1.1 ADS State of the Art

In this section, typical ADS architectures will be dealt with for both civil and UAV applications and the main differences between the systems will be highlighted. Firstly, ADS for current civil aircraft will be presented, where the IAS (or CAS), TAS, barometric height and Mach number are strictly necessary for navigation and control. Although the angle of attack is measured, this information is not usually given on pilot displays but is only used by the stall prevention system to warn pilots of oncoming aircraft stall. Secondly, the ADS for UAV applications will be presented, where the IAS, TAS, Mach number, barometric height, and the angles of sideslip and attack are required by the flight control system for automatic control. In order to have an idea of how the ADSs could be designed, several configurations will be shown using realistic but not real architectures. They are in fact only intended for comparison purposes between the advantages and disadvantages of designed architectures using both off-the shelf and state-of-the-art sensors.
A simplex (not redundant) standard civil ADS configuration for stable aircraft is depicted in Fig. 1.2. The word *standard* when referred to ADS is used to

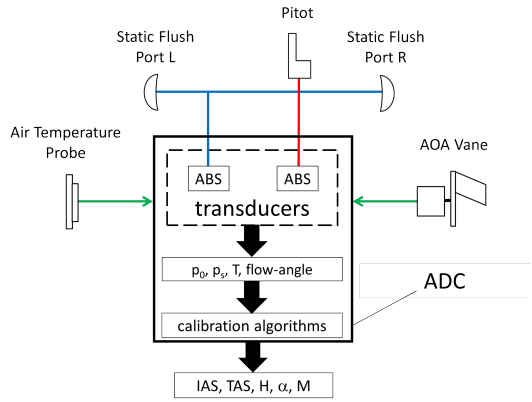


Figure 1.2: A realistic standard ADS architecture for civil aircraft. The red lines represent the total pressure lane, the blue lines the static pressures that are actually connected and the green lines represent the electrical signals

indicate the use of off-the-shelf sensors and ADC. The Pitot tube measures the total pressure, while the left and right flush ports sense the local static pressure and they are pneumatically averaged together to minimize sideslip effects (tipically it is an adequate solution for $\beta < 10$). Generally, the angle of attack, α, is measured

by means of vanes, while the air temperature is measured using dedicated static air temperature (SAT) or total tir temperature (OAT) probes. Moreover, as previously mentioned, although the angle of sideslip, β, is not required for the most of civil aircraft, it could be measured, for example, using an additional vane. All the measured data are converted into electronic signals and then processed by the ADC in order to be converted into known quantities, such as the IAS (or CAS), TAS, barometric height, Mach number and angle of attack,α, and in just few cases the angle of sideslip, β, is also measured for military or unstable aircraft.

Some simplifications can be introduced into the standard architecture shown in Fig. 1.2, if state-of-the-art or *advanced* probes are used instead of standard ones, as depicted in Fig. 1.3. Basically, advanced probes are multi function probes that are able to measure three quantities using only one external sensor: total pressure, static pressure and one or two flow angles. Several models exist, such as the integrated Goodrich *SmartProbe*© used on some regional Embraer business aircraft and on Airbus A400M, the *MFP* (which is a self-orienting cone) manufactured by Aerosonic Corp. used on Alenia Aermacchi M346 and the self-orienting probe manufactured by Thales (which is a flow angle vane with pressure ports for total and static pressure sensing) used on the Dassaul *Rafale* and on EF2000 (licensed to Marconi Avionics by Thales). All these probes are integrated with an electronic box (Air Data Unit, ADU) that incorporates: pressure and position transducers, as well as computing and self-test capability thus eliminating the need for a central ADC and the need for pressure tubing running along the airplane, as described in Fig. 1.3, and which will be referred to in this work as *advanced* ADS. Each aircraft, according to its design specification, is provided with a specific ADS architecture. Some military aircraft, for example, have other state-of-the-art designs. Another state-of-the-art architecture, which exploits nonobtrusive sensors, is in fact used by hypersonic, stealth, and some research aircraft, in order to meet their particular requirements. This system is known as Flush Air Data System, FADS [23, 24]. It consist of multiple flush pressure ports distributed over the aircraft surface, and on the nose of conventional aircraft [25]. The pressures can be processed by an ADC or by an FCC to obtain a complete set of air data, as shown in Fig. 1.4. This function

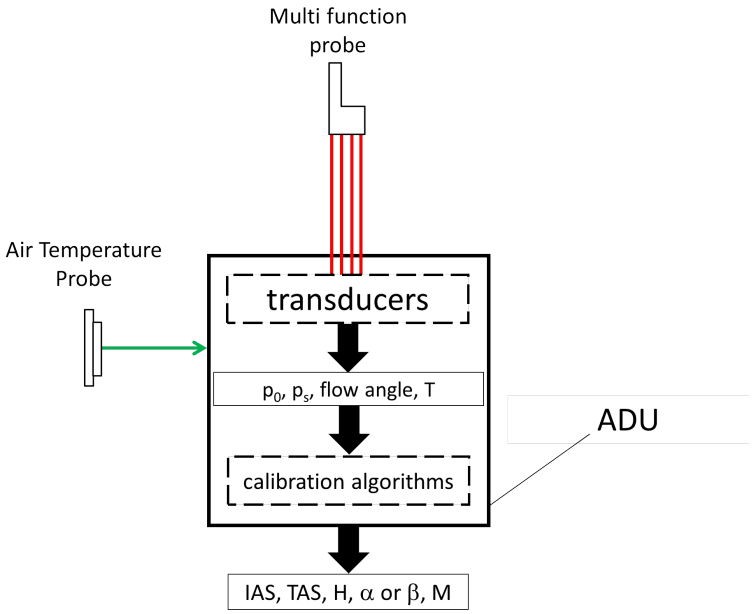


Figure 1.3: A possible *advanced* ADS architecture for civil aircraft. The red lines represent the electrical output signals

can be carried out using equations for potential flow around a sphere and then correcting them for nonpotential and nonspherical flows with calibration coefficients. Generally, FADS is an overdetermined system: it measures more pressure than is actually required for the air data state. Inaccurate pressure readings can be excluded through calculations, and its robustness can thus be improved with respect to the other ADS presented in this section. Moreover, FADS is used on some special aerial-space research vehicles, such as the Lockheed Martin X-33 [26], which use their own calibration algorithms to convert the raw pressure measurements into the requested set of air data. In the future, another class of nonintrusive sensors may be used, that is optical airdata sensors. These sensors use lasers to measure the speed, flow angles and temperature of the air. They only need a flush window in the airplane for the laser equipment.

Unitl now, ADSs for civil and military aircraft have been presented for piloted airplanes, but the following *complete* suite of air data is required for the automatic control and navigation of any kind of manned or unmanned, stable or unstable:

$$\{p_0,\ p_s,\ T_s \text{ or } T_0,\ \alpha,\ \beta\},$$

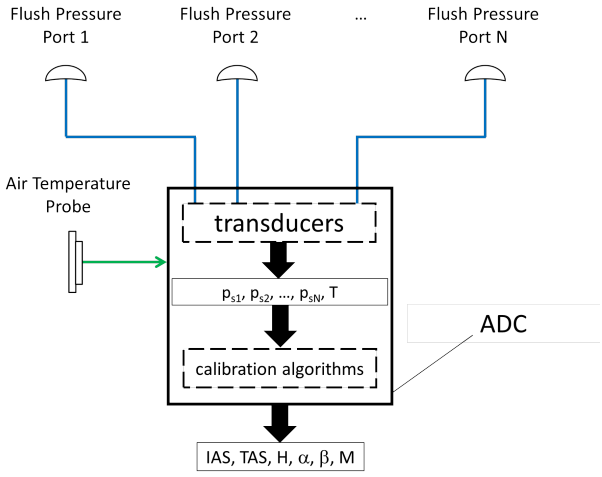


Figure 1.4: FADS architecture. The green line represents the electrical output signal and the blue lines represent static pressures lanes

where free stream values of aerodynamic angles are indicated as α and β for simplicity. As far as unmanned aircraft are concerned, the complete air data set is necessary to fly safely along the entire flight envelope, from take-off to landing. Nowadays, UAVs are usually equipped with an air data boom (shown in Fig. 1.5), which is able to provide a complete set of air data. It can be built using common nose booms or five-hole probes [27]. Air data booms are usually used on UAVs as the primary and

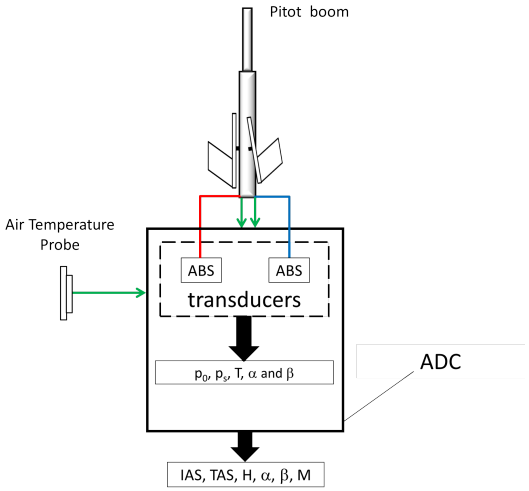


Figure 1.5: A standard ADS boom architecture for UAV aircraft. The red lines represent the total pressure lane, the blue lines the static pressures that are actually connected and the green lines represent the electrical signals

the sole ADS because of their capability of measuring free stream air data, without requiring calibration algorithms. Because of the last two reasons, this kind of ADS

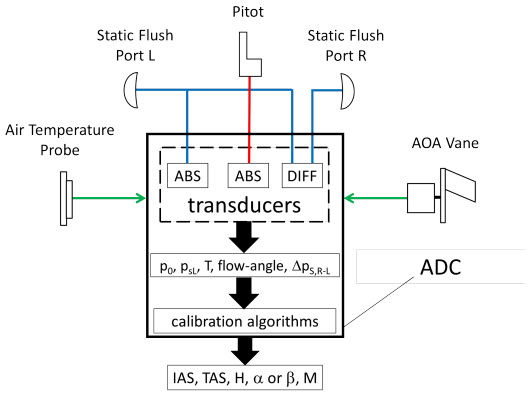


Figure 1.6: A possible ADS architecture for automatic control and navigation using *standard* probes. The red lines represent the total pressure lane, the blue lines the static pressures that are actually connected and the green lines represent the electrical signals

is often used as a calibrating system (see section 1.2) and hence is indicated as a *flight test* air data system. Although the advantages ot the air data boom solution are well known, the potential issue associated with intrusion into the field of view of forward looking cameras, makes this solution not completely suitable for use on UAVs. In addition further complexity is added if a multiple nose probe arrangement is required.

A possible ADS for unmanned flight vehicles, based on *standard* probe technology, is shown in Fig. 1.6 and it is very similar to that presented in Fig. 1.2, but another static pressure sensor was added which allows the angle of sideslip, β, to be calculated using the difference between the left and right static pressures, as in (1.1). Even though this architecture (the group of probes, sensors and ADC) has the enormous advantage of being economic, it needs several external sensors mounted onto the fuselage, and hence it is not very convenient to redound this system for modern UAVs.

The last ADS architecture, presented in Fig. 1.6, can be simplified to a great extent using advanced probes. A possible advanced ADS, which is able to provide a complete set of air data, is represented in Fig. 1.7. In the latter case, the multi function probes are adequately mounted onto the aircraft on opposite sides of the nose to correctly sense the angle of attack, in addition to the left and right static pressures and two total pressures. Even though the redundant measurements of α

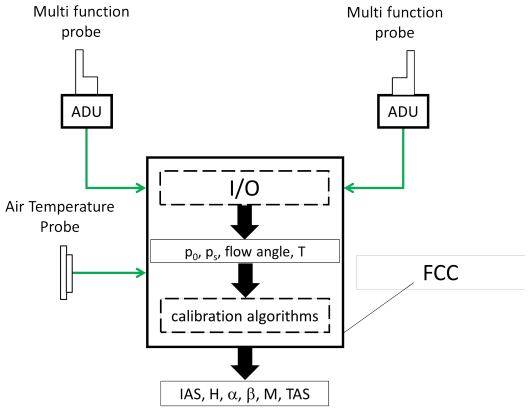
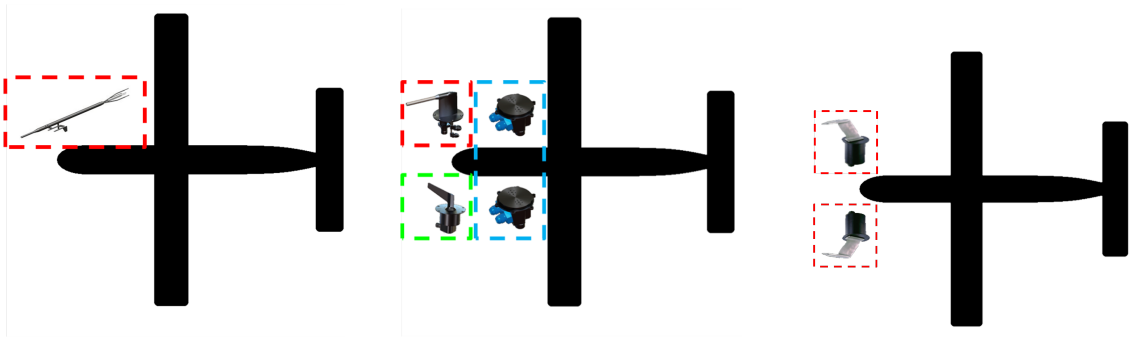


Figure 1.7: An advanced ADS architecture using multi-function probes. The green lines represent electric signals



(a) Air data boom (b) *Standard* ADS (c) *Advanced* ADS

Figure 1.8: Possible ADS architectures considering current air data technologies

and P_0, are not required for a simplex system, the two static pressure measurements are necessary to correctly calculate the static pressure in the presence of lateral wind and to use in (1.1) in order to calculate the angle of sideslip, β. Considering that the ADS costs are just considered here in a qualitative way, it is worth mentioning that it is always more expensive when advanced probes are used in ADS, than those with standard probes shown in Fig. 1.6, but this allows the initial 5 external probes to be reduced to 3, two multi function integrated probes and one temperature sensor. In order to summarize the possible ADSs for UAVs and to highlight the differences, Fig 1.8 describes three possible ADSs identified here according to current technology: air data boom, classic or advanced. A temperature sensor does not appear because each system is edowed with one. In order to comply with current safety requirements, one of the ADS represented in Fig. 1.8, which is able to provide the complete set of air data, needs to be physically duplicated, or even triplicated, to allow UAVs

to fly in a controlled airspace in order to carry out aerial work. The air data boom in Fig. 1.8(a) is not suitable for redundancy purposes mainly because of the interference with electro-optical sensors, as previously mentioned. Although the ADS, which exploits the standard probes shown in Fig. 1.8(b), can easily be duplicated, it requires a large number of external probes with the consequent issues regarding the mounting positions. Considering the complexity of an ADS system, the use of advanced probes (Fig. 1.8(c)) allows to redound the system with fewer external probes than the standard ADS, but with higher costs. As usually occurs in the engineering field, there is no best solution but rather a best compromise: the redundant ADS design is always a compromise between technical requirements, performance and costs.

A software-based virtual sensor will be introduced within this scenario in order to replace physical sensors and save on hardware with the benefits discussed in chapter 2.

1.2 ADS Calibration

In all the described cases, the ADS probes need to be calibrated to obtain free stream data. The probes located on the aircraft surface skin perform measurements, commonly indicated as *local* measurements, which are affected by errors due to the location of the sensors. Moreover, the presence of the aircraft and its motion distort the external flows from the free stream conditions. The resulting error from the aforementioned contributions is simply indicated as the *local-to-true* error, whereas when it is refers to static pressures (and also dynamic pressure) it is defined as *position error*. In order to remove the errors due to the location of the probes, ad-hoc calibration laws are introduced to convert the local sensor measurements into the free stream ones.

The most sensitive-to-location air data is the static pressure, because of the considerable pressure flow distribution changes that takes place along the fuselage, see Fig. 1.9 from [22], which are functions of the aircraft shape, flight regime (Mach number) and aircraft attitude (α and β). While the total pressure measurements do not

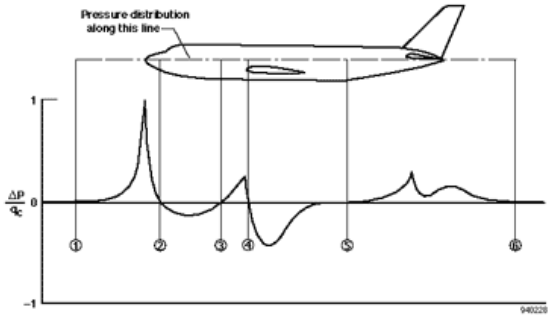


Figure 1.9: A typical pressure distribution along the fuselage

depend significantly on the location of total pressure probe, the static pressure is highly influenced by the position error affecting the calculations of dynamic pressure and of barometric altitude which is used to vertically separate aircraft in airways worldwide. It is indicated in hundreds of feet, for example, an aircraft flying at 33000 ft, is indicated with the international standard $FL330$. The static pressure is usually calibrated using a correction factor $\Delta C_p = \frac{p_{s,i} - p_{s\infty}}{q_{c\infty}}$ as a function of the Mach number (see Fig. 1.10), while considering, at this stage, only a very slight influence of the aerodynamic angles. As far as low velocity is concerned, larger C_p are acceptable according to aviation regulations, as described in [28], in fact a $\pm 30 feet$ error can be accepted for Mach numbers below 0.4 ($M < 0.4$) in the landing configuration.

The total pressure is only influenced slightly by $Mach$ numbers over 0.9 and by angles of attack, and also sideslip, higher than 10 deg; corrections are required outside these boundaries. The air flow angle vanes must be calibrated in order to remove errors that derive from the influence of fuselage aerodynamics and the Mach number. The temperature measured by the static temperature probes is higher than the free stream air temperature because of frictional heating, radiation from the thermometer to the external environment; total temperature probes measure higher temperatures due to the same issues regarding the static temperature probes and also for the non-isentropic compression of the air on the total temperature sensor. A recovery factor is usually adopted to correct the temperature probe measurements and it is a function of the Mach number over 2 and the angle of attack over 10 deg. Several calibration techniques exists to calculate the compensation factors for each

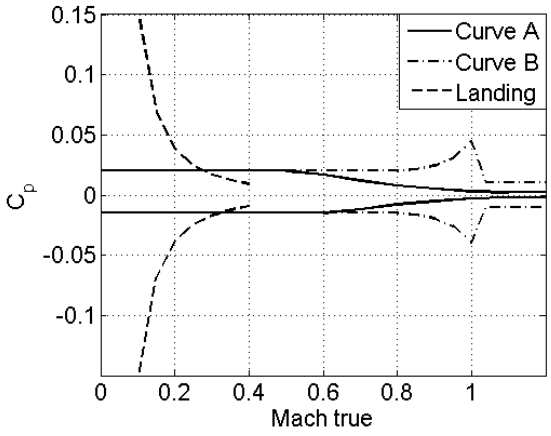


Figure 1.10: Position error tolerance [28]. If the position error is entirely within the envelop of curve A, then no correction is required. If the position error falls within the envelop of curve B, a compensation is required to make the position error in within the curve A. If it falls outside the curve B, a new mounting position is suggested. The landing profile regards flight regimes for $M < 0.4$

of the air data.

The most commonly methods, which are presented in the next two sections, can be split into two main groups: indirect and direct. Tower flyby is the most reliable indirect calibration technique for pressure position error compensation. Among the direct methods, the flight test boom is the most accurate and simplest technique to correct all the air data measurements. It is in fact widely used in the UAV field because of its time and cost saving characteristics. The following description of methodos used for calibrating ADSs, is here reported in order to describe how sophisticated are the classical calibration methods and to highlight benefits and drawbacks of all the methodologies presented here.

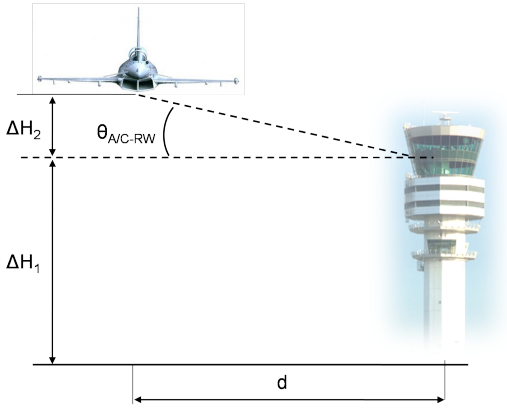
1.2.1 Indirect Methods

Three indirect methods for pitot-static system calibration will be presented in this section. The first is the tower flyby method, which is mainly used for static pressure data calibrations. The second is the radar tracking methodology, which represents a way of extending tower flybys to supersonic velocity. The third methodology is a further extension of the tower flyby method. It is obtained by performing several dynamic maneuvers, which are tracked by radar, to compare the measurements with

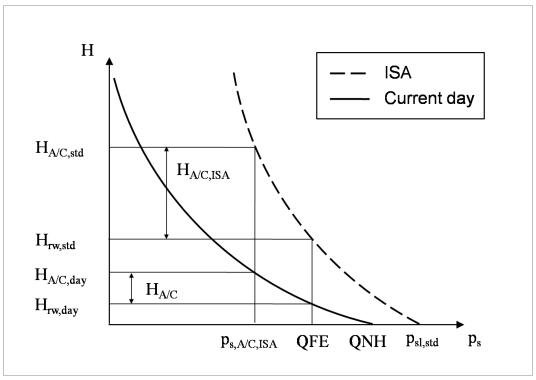
those carried out by the air data system being tested.

Tower Flyby

This method is extensively used for static pressure and altimeter calibration because of its high accuracy. Tower flyby is an indirect calibration methodology, because the test static pressure is calibrated by comparing with another one calculated at aircraft height starting from the pressure and temperature measured at the ground level of a runway on the basis of the standard lapse rate. The aircraft, equipped with the ADS being tested, is flown along a prescribed path over the runway, at constant velocity, at a height of about 100 ft. The flight is recorded using a visual indicator, such as a sighting stand or a phototheodolite, and the resulting angle, as shown in Fig. 1.11(a), is used to calculate the real, or geometric, aircraft altitude, $H_{A/C} = \Delta H_1 + \Delta H_2$. All the quantities in the tower flyby method must be referred to the international



(a) Geometric aircraft height, $H_{A/C}$



(b) International Standard Atmosphere (ISA) model and current weather conditions

Figure 1.11: Description of tower flyby method

standard atmosphere (ISA) in order to avoid referring the calibrations to current day weather conditions, for example, when temperature and pressure at sea level are lower than standard values, as depicted in Fig. 1.11(b). In fact, in (1.2), the measured geometric aircraft height, $H_{A/C}$, is corrected to convert the measurement to standard conditions, $H_{A/C,ISA}$, in order to refer the pressure calculated in (1.2) to standard conditions. Therefore, the static pressure used as a reference in the tower flyby calibration is calculated starting form the static pressure measured at

the runway level, QFE, as

$$p_{s,A/C,ISA} = QFE\left(1 + \frac{\beta}{T_{s_{rw}}} H_{A/C,ISA}\right)^{-\frac{g}{R\beta}}, \tag{1.2}$$

where $\beta = 0.0065\ \dfrac{K}{m}$ is considered constant inside the troposphere ($H \leq 36000\ ft$) and $H_{A/C,ISA}$ is

$$H_{A/C,ISA} = \frac{T_{rw,ISA}}{T_{rw,day}} H_{A/C}. \tag{1.3}$$

A graphical approach of the calibration phases of the tower flyby method is reported in Fig. 1.12 in which the route from the measured QFE to the reference static pressure, $p_{s,A/C,ISA}$, is described by also exploiting the aircraft height referred to standard weather conditions, $H_{A/C,ISA}$, and calculated using (1.3).

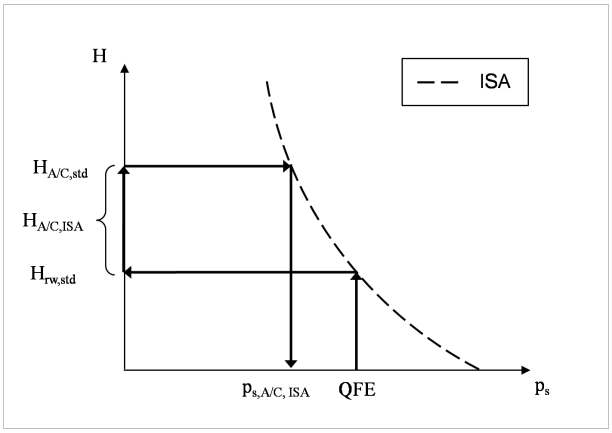


Figure 1.12: Process to calculate the reference static pressure at aircraft height referred to standard weather conditions

The tower flyby methodology can be exploited from slightly higher velocities than stall, usually higher than $1.3V_s$, for safety reasons, to the transonic regime ($M < 0.85$), to avoid any damage to civilian buildings. A single tower flyby flight lasts a few minutes, but is repeated at several velocities to cover the entire flight envelope; sometimes a single point (or velocity) is repeated more than once in order to collect more data to minimize the measurement errors. In certain circumstances, data collected during tower flyby tests can also be used for a preliminary correction of the angle of attack at subsonic Mach numbers. In fact, since the velocity is constant, the analytical free-stream angle of attack can be calculated using the lift coefficient expression and used to correct the angle of attack measured from the air data system aboard.

Radar Tracking

A similar method to tower flyby is that of the radar tracking calibration method, although it is less accurate. It is in fact often used to extend the position error compensation obtained using the tower flyby method to higher velocities than the speed of sound ($M > 1$) for supersonic aircraft. The airplane is accelerated, usually over the sea or very far from civilian buildings, till a maximum velocity is reached, while a constant heading and geometric height are maintained and measured using radar. The frame within the flight data collected during acceleration, in which the Mach number, or $q_{c_{A/C}}/p_{s_{A/C}}$, is equal to one determined during the tower fly by is singled out. The position error correction, C_p, is known at this time frame from the tower flyby calibration and is considered the same for the accelerated flight. Hence, the real or calibrated barometric height can be calculated at this frame. The last corrected barometric height is used to correct all the other altitudes recorded during the accelerated flight, and the position error correction can be calculated for all the other time frames using the (1.2).

Dynamic Maneuvers

The radar tracking calibration method can be extended at higher altitudes, exploiting a radar system, or other Earth-relative data sources, to reconstruct the trajectory of dynamic maneuvers, which are corrected from external environment disturbances using weather analysis in order to obtain the free stream data. Descents and climbs are performed for baro-altimeter and, hence, for static pressure calibration purposes: the altitude measured on board of the test aircraft is compared with the altitude calculated using radar or other considered devices.

1.2.2 Direct Methods

Three direct calibration methods, which are commonly used to calculate the position error correction for pressure measurements, will be presented in this section. Aerodynamic angles can instead be corrected only using the flight test boom technique.

Flight Test Air Data Boom

The most commonly used direct calibration method exploits the pitot boom (see Fig. 1.8(a)), which is usually equipped with two flow angle vanes that are used for free-stream aerodynamic angle measurements. The flight test boom calibration method is used to calibrate all the quantities of the previously presented air data set and can be used at any Mach number. The aim of this calibration methodology is to calibrate ADSs in the presence of severe dynamic maneuvers. Windup turns are performed to obtain data at elevated normal forces or angle of attack, while roller coaster and pushover-pullups are used for angle of attack calibration. Rudder sweeps are performed for angle of sideslip calibration purposes.
The length of the boom probe is a related to the disturbance introduced by aircraft in the external flow. Although a very long boom could be an optimal configuration for free stream air data collection, dynamic instabilities, which can occur at certain velocities and attitudes and which introduce noise into the measurements, due to vibrations, can limit the length of the boom. An adequate strut is in fact designed for the required boom length.

Trailing Cone

The trailing cone method is another direct-comparison type of calibration. A long tube behind the aircraft can measure air data at nearly free-stream conditions using an adequate probe. The perforated cone acts as a drag device, which, coupled with a tube of varying length, is able to stabilize the end-tip probe (see Fig. 1.13(a)), since the optimum extension length to guarantee trailing cone stability depends on velocity. In order to cover the entire aircraft flight envelope, an extending/retracting system must be designed to avoid any dynamic instability of the tube. This calibration method is only used for static pressure correction and, with respect to the tower flyby method, has the advantage of being able to validate the tower flyby calibration and extend it to higher altitudes.

(a) Trailing cone method (b) Pacer aircraft method

Figure 1.13: Two direct methods to calibrate the static pressure

Pacer Aircraft

This direct-comparison calibration method involves a second airplane, known as the pacer aircraft, which is used as a free-stream source of pressure data. An accurately calibrated air data system aboard the pacer aircraft is used to calibrate the test airplane. Both aircraft fly at nearly the same attitude and altitude and if any difference exist, an altimetry correction is carried out using the altitude difference measurements. Unfortunately, this method suffers from the drawback that the calibration accuracy cannot be better that the accuracy of the air data system of the pacer aircraft.

1.2.3 Matlab Code for Calibrating ADS

Two methods will be presented in this section, that were developed during this research project to calibrate ADS, in particular the total pressure and aerodynamic angles using a direct method which exploits reference air data from a pitot boom, while the static pressure is calibrated using the indirect tower flyby method. The routines are aimed for UAV application, but can also be used for manned aircraft.

Tower Flyby

The Matlab tower flyby routine, called *CaliTow*, is essentially a code which summarizes the internal procedure of Alenia Aermacchi.
The code needs the following inputs from flight test conditions:

- pressure at the runway level (QFE),

- temperature at the runway level,
- altitude of the runway,
- any known bias of the pressure transducer.

From a graphical user interface (GUI), the user is requested to provide different information recorded during flight tests:

- geometric altitude from the runway,
- indicated static pressure of the ADS that has to be calibrated,
- indicated dynamic pressure of the ADS that has to be calibrated,
- indicated Mach number of the ADS that has to be calibrated.

The routine then calculates the compensation factor of the ADS tested for static and dynamic pressure, according to method presented in 1.2.1. At this stage, the corresponding true values of any local pressure measurements are known for each time step of the flight data. Therefore, a new file is created from the initial flight test data file, in which new columns are added for the true, or free-stream, static pressure, dynamic pressure (considering the total pressure to essentially be correct), CAS and M_T data. From the entire flight test data (about one hour of data recording) only a few points (about 10) are manually identified as the most representative. These reference points are usually extracted during the stabilized flight conditions when the accelerometers and gyroscopes are only slightly excited. A new file is then generated automatically, with the previously chosen representative points, in which both the indicated and corrected values of p_s, q_c, CAS and M are contained. The user is then asked to choose the order of the pressure error correction (PEC) polynomial, which, in this treatment, is only a function of indicated Mach number,

$$C_p = \frac{\Delta P}{q_{c,i}} = \frac{p_i - p_\infty}{q_{c,i}} = f(M_i). \tag{1.4}$$

The calculated PEC could easily be implemented, as depicted in Fig. 1.14, in ADC as static pressure.

Generally speaking, the PEC depends on various parameters, $f(p_{s,i}, q_{c,i}, \alpha_i, M_i)$, but in the present work all the other dependencies have been considered negligible with

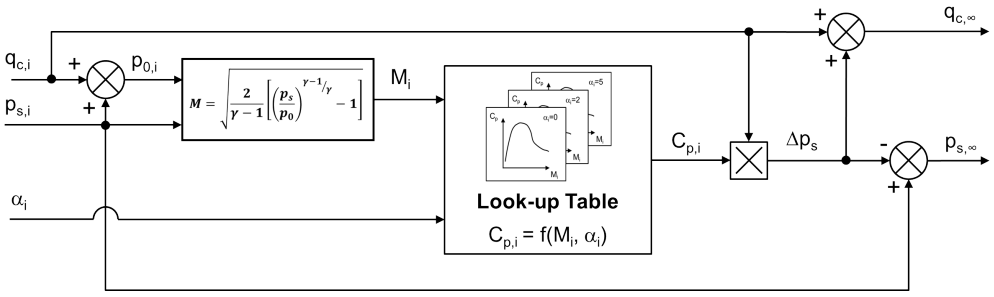


Figure 1.14: Diagram block of PEC implementation in ADC

respect to that of the Mach number. The described procedure is often used without a calibration polynomial, but look-up tables are adopted, which are able to take into account more dependency parameters than only the Mach number, such as the angle of attack, sideslip and so on.

Since the dynamic pressure is calculated as the difference between the total and static pressure, the total pressure is sometimes considered correct, or it is corrected using direct methods, because it is influenced less by position errors. Under this hypothesis, once the static pressure is corrected, the dynamic pressure is also corrected. Once the PEC is obtained, the resulting look-up tables are implemented in the process for static and dynamic pressure calibration, see Fig. 1.14.

Flight Test Air Data Boom

The here presented Matlab routine, called *CaliRef*, is essentially an interpolating code which minimizes the mean squared distance between local and free stream data.

Just several points (about 10) are manually identified from the entire flight test data (about one hour of data recording) as the most representative of the stabilized flight conditions with the corresponding time steps. These reference points are usually extracted when accelerometers and gyroscopes are just slightly excited.

As the time step list is formed, it is inserted into the Matlab routine, which extracts new columns from flight test data with the corresponding free stream (from the boom) and local measurements (from the test ADS) of the total pressure, angle of attack and sideslip. At this point, the polynomial order for each of the quantities is selected and the optimization routine returns the requested polynomial coefficients.

Chapter 2

Virtual Sensor Design

In this chapter, the virtual sensor will be presented, from its first concept till its final version. The virtual sensor concept arose after some ADS multi-probe architectures proposed for MALE (Medium Altitude Long Endurance) applications were studied and realized that a completely satisfactory architecture was not available on the market for this kind of aircraft, due to the intrinsic complexity when redundant systems are required. Some technological issues do in fact exist, such as the need to keep costs low, which is an important driver on the UAV market, but which is often in conflict with the need to introduce innovations and the need to keep the aircraft weight down, in order to increase the payload capability in terms of weight and volume, and this is usually in opposition to certification regulations, which require redundant systems for safety reasons. As far as ADS is concerned, many other issues also exist. Let us consider, for example, that most ADS sensors need to be installed in the forward part of a UAV fuselage, which is already taken up by the payload, avionic equipment, radomes, antennae and opto-electronic sensors specific for the aircraft mission. Another important issue concerns the so-called *bird strike* event. Because of the reduced size of UAVs, it can sometimes be a problem to assemble several external air data probes or sensors adequately spaced out to comply with *bird strike* certification. Therefore, considering these, and other minor issues, the need to reduce the number of actual sensors seemed to be of primary importance: the virtual sensor concept thus arose. Since both total and static pressure may be calculated using standard Pitot-static tubes, a remarkable simplification of the ADS architecture (see Fig. 2.1(a)) can be obtained through the use of virtual sensors for

aerodynamic angle estimation. Hence, in this work, a virtual sensor was conceived for aerodynamic angle estimation. It was also thought up to be used both as a primary source of data and as a stand-by system which could be used for voting and monitoring purposes, in order to lower the level of redundancy.
At the beginning, very few design requirements were set for the sensors:

- it had a comparable performance as with current actual sensors: maximum errors within ± 1 *deg*;
- it had to run in a few milliseconds on real FCC, in order to work in real time;
- it had to be able to use all the available data at the FCC as other inputs.

On the basis of experience gained in Alenia Aermacchi, a tolerance band of ± 1 *deg* was defined which stemmed from three main contributions: errors of the current angle vanes, which is within ± 0.4 *deg*; errors due to the calibration algorithm, which is within ± 0.3 *deg*; errors caused by the installation, which is within ± 0.3 *deg*. Each of the three error contributions could be expanded into sub-contributions, but it would be beyond the goal of this work.
The most important novelty introduced by virtual sensor is the indirect measurement of aerodynamic angles unlike all the other sensors built till now which measure aerodynamic angles in a direct way adopting some pressure measurements using air data probes from the airflow surrounding the aircraft. Therefore, an innovative way to estimate aerodynamic angles was required to replace the physical sensors by exploiting the flight data already measured on-board the UAV, such as using inertial data. The choice between a model based technique or neural network will be dealt with in the next sections.

2.1 Project Requirements and Objectives

A NN-based processing system was proposed as it was easier to implement and was inherently stable alternative choice compared to model based techniques.
As far as the mathematical model based technique is concerned, an aircraft model is required to define the state vector, its derivative, and hence its ODEs that have to

be solved to obtain the desired measures of aerodynamic angles. The modeling of a very complex system, like an aircraft, will always introduce some discrepancies from the real-world system. Building a reliable aircraft mathematical model can in fact be considered a mirage: an aircraft model and its subsystems are usually approximated and may require several tuning iterations after comparisons with actual flight test data. Under these conditions, neural networks may be more advantageous than model based techniques. In fact, when neural networks are used, it is possible to model a real-world plant without knowing anything about the dynamic model, just by training NNs with the observed input and output patterns. It is obvious that the knowledge of the analytical equations of actual systems, which which can be used to describe their working and evolution, can be a great help to engineers to obtain a better design of neural networks.

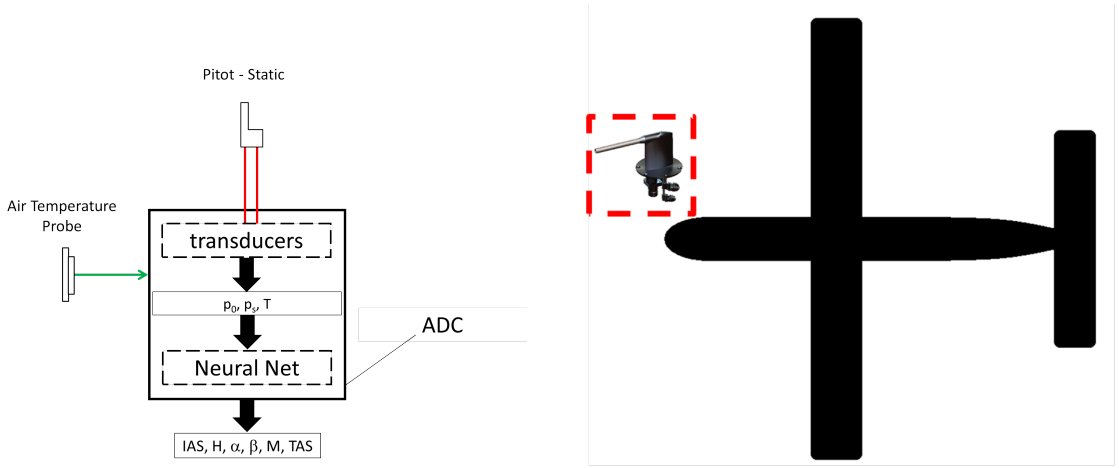
However, even though a very precise mathematical aircraft model is available, the virtual sensors would need to be run in real time mode: indirect measurements of aerodynamic angles require the aircraft model to run in a very short time, of the order of magnitude of milliseconds, on the actual FCC, which is slower than modern personal computers. First of all, this issue seemed an enormous problem needed to be solved, using a model based technique. Moreover, programming a mathematical model on aircraft FCC is not a simple task, due to the time consuming recursive methods that are needed to implement the mathematical model, and for safety reasons which always suggest avoiding sub-iterations to converge at each time step, when possible. Regardless of how complicated neural networks are, they can always be reduced to several matrix calculations with the use of non-linear functions. Therefore, neural networks overcome the drawback of *time consuming software* and the presence of inner loops.

The main drawback of neural networks compared to model based techniques is the meaningless of the network coefficients (synaptic weights), while aircraft mathematical model coefficients have a physical meaning. This is a very important aspect of virtual sensors in real life operations. Let us consider, for example, changing the aircraft center of gravity (CG) by simply relocating the payload. In order to maintain the same virtual sensor performance, the neural network should be retrained with the

new weight and balance configuration. Instead, if a mathematical model is used, changing the *cg* is a very simple operation. It is clear that even very slight changes in the aircraft configuration could decrease the virtual sensor performance and, this requires investigations using a simulator, to avoid unexpected errors.
The proper construction of a neural network requires a training set to be developed which would accurately and adequately represent the plant which has to be learned, as it will be described in section 3.5. For this purpose, several maneuvers are required at different speeds in all flight conditions (flap extension, still or turbulent air, and so on) in order to cover the entire flight envelope. Obviously, this is not realistic, and the number of maneuvers, and thus training data points, will be reduced so that they can be managed by a common workstation, as will be described later on. Furthermore, the optimum architecture and degree of training still has to be determined heuristically during the training process.
The conclusion of this preliminary *technique selecting* activity was that the characteristics of neural networks are better suited the initial requirements better than a model based technique and were therefore selected as the technique on which to build the virtual sensors.

2.2 Trade-off: Advantages and Drawbacks

All the presented ADS solutions presented in section 1.1 are realistic layouts considering the sensors available on the market. Since the basic set of air data (see section 1.1) is required in any aircraft equipped with autonomous control system, the issue is how to design the ADS according to the desired type of sensors. An analysis on each system, presented in section 1.1, will be carried out in this section in order to highlight some of the advantages and drawbacks of each one, and better compare the virtual sensor with the actual ones.
The system depicted in Fig. 1.6 has the enormous advantage of being made up of a very consolidated technology and of being available on the market. The main drawback is the number of external devices that are necessary with the consequent complexity due to wirings, piping and mechanical constraints.



(a) ADS with neural networks

(b) Possible ADS architecture using NNs

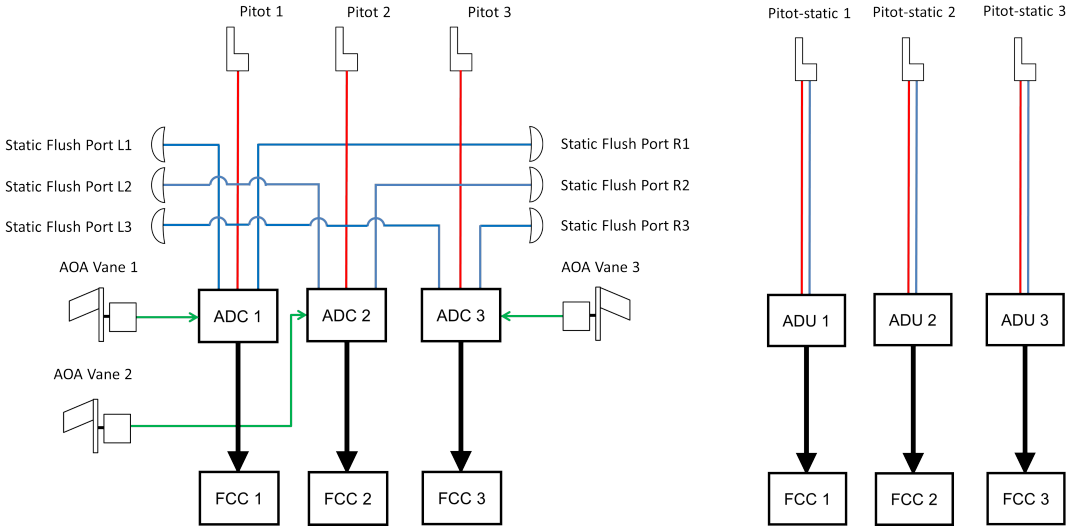
Figure 2.1: A possible installation of ADS probes using virtual sensors for aerodynamic angle estimation based on neural networks

The advanced ADS presented in Fig. 1.3 allows engineers to save two external sensors, but most of the sensors need to be installed in the front part of the fuselage, which could be a problem for some UAVs. Even though a cost evaluation is beyond the scope of this work, it is clear that such a system could be more expensive than the first one.

Since FADSs (described in Fig. 1.4) are more complicated than the other architectures mentioned here, they are only suitable for very particular application where a flush system is the only technological solution that can be adopted. The virtual sensor can be used in a realistic architecture, as depicted in Fig. 2.1(a), and can introduce the following advantages compared to other modern architectures:

- a saving of external actual sensors and the related costs;
- a weight saving;
- an onboard saving;
- a reduction in maintanance: fewer items means improving ADS reliability.

Moreover, it is clear that the ADS in Fig. 2.1(b) can be easily redounded if compared with all the other possible current ADSs represented in Fig. 1.8. In fact, as shown in Fig. 2.2, a realistic triplex ADS architecture can be simplified to a great extent introducing virtual sensors for α and β, and at least nine external probes are thus



(a) Triplex ADS using off-the-shelf air data probes

(b) Possible triplex ADS architecture exploiting NNs

Figure 2.2: Realistic triplex ADS architecture using, or not, virtual sensors for aerodynamic angles estimation based on neural network.

saved if compared with ADS based on off-the-shelf probes: six static flush ports and three α vanes. Moreover, another advantage of using virtual sensors is that no dedicated ADC is needed. Indeed, The ADCs in Fig. 2.2(b) were substituted by ADUs, which are cheeper than the more complex ADCs because they do not offer any calculation capabilities but only a set of transducers.

Overall, as far as analytical redundancy is concerned, a virtual sensor, based on neural networks, has been identified as the best strategy to reduce complexity inherent current redundant ADS sensors.

Chapter 3

Neural Network

3.1 Historical Background

In 1955, McCarthy et al. [29] coined the term artificial intelligence (AI) to identify that branch of computer science in which some techniques share the ability of the human mind to reason and learn in an environment of uncertainty and imprecision. The aim of AI is to create intelligent machines that mimick human intelligent behavior by expressing it in language forms or symbolic rules [30, 31]. Within the computer science group, *soft computing* (SC) belongs to the artificial intelligence branch. According to Zadeh [32], soft computing has the aim of adapting to the pervasive imprecision of the real world, unlike traditional, or hard, calculation methods. The driving principle of SC is to "exploit the tolerance for imprecision, uncertainty and partial truth in order to obtain tractability, robustness and low cost solutions". Soft computing includes three human inspired techniques: the artificial neural network (ANN) or simply the neural network (NN), fuzzy logic (FL) and genetic algorithm (GA). GA is a heuristic search that mimics the process of natural evolution. The fuzzy logic is based on calculation using linguistic labels stipulated by functions, called membership functions. A selection of if-then rules are the core of the fuzzy inference system, which can model human expertise in several specific applications.

Inspired by the human brain structure, neural networks are expected to mimic brain mechanisms to simulate human expertise in decision making simply by using matrix calculations. In 1904, Cajal [33] introduced neurons as the basic component of the

human brain; since then many researchers have been involved in mathematically reproducing the complex, nonlinear and parallel computer that the human brain is. After a few decades, several artificial intelligence techniques were completed and fully demonstrated, but owing to the high calculation power that they required, they did not spread very much. The first mathematical neuron model called the perceptron model, was proposed by Rosenblatt in 1958 [34]. This model is still widely used in the neural network field, but a rigorous demonstration required about forty years and was only proposed after the neural network had been successfully used. The fundamental equation of feed-forward neural network units (reported in (3.1)) is very similar to the expression proposed by the neurophysiologists McCulloch and Pitts [35], already back in 1943, to mimic nervous activity. Only starting from 1987 did the authors, Hecht-Nielsen [36, 37], Lippmann [38] and Spreecher [39], suggest that Kolmogorov's theorem (1957), concerning the realization of arbitrary multivariate functions, could provide theoretical support for neural networks.

Theorem 1. (Kolmogorov's Theorem) *Any continuous real-valued functions* $f(x_1, x_2, ..., x_n)$ *defined on* $[0, 1]n$*, with* $n \geq 2$ *can be represented in the form*

$$f(x_1, x_2, ..., x_n) = \sum_{j=1}^{2n+1} g_j \left(\sum_{i=1}^{n} \Phi_{ji} x_j \right),$$

where the g_j*'s are properly chosen continuous functions of one variable, and the* ij*'s are continuous monotonically increasing functions independent of* f*.*

Other authors, such as Girosi and Poggio [40], instead criticized this interpretation of Kolmogorov's theorem and stated that it was irrelevant for neural networks as the Φ_{ij} functions are highly non-smooth. As this debate continues, the importance of Kolmogorov's theorem has pointed out the feasibility of using parallel and layered network structures for multivariate function mappings, and of not proving the universality of neural nets as function approximators. Other authors, Cybenko [41], Hornik et al. [42] and Funahashi [43], independently proved analytically that one hidden layer feed-forward neural network is able to uniformly approximate any continuous multivariate function, through the use of continuous sigmoid functions (instead of non-smooth Φ_{ji}), as well as other more general activation functions. Since the late 1980's, thanks to computing progress, several complex problems have been

solved using neural networks developed on personal computers, but the performance of the human brain still remains a mirage. In fact, although silicon gates are six orders of magnitude faster than human neurons, the human brain is able to process a large amount of information much faster than modern computers.

3.2 Theoretical Background

In 1987, the neuroscientist and computer scientist Arbib [44] described the nervous system depicted in Fig. 3.1, where the human brain continuously receives information from surrounding environment (*stimulus*) and makes decisions (*response*). The two sets of contrary arrows indicate that the brain can communicate with the receptors and effectors; the arrows pointing right indicate the forward transmission, while the others indicate feedback transmission. It is clear that the response could also be sensed by the brain and then used as input. Continuing the parallelism between

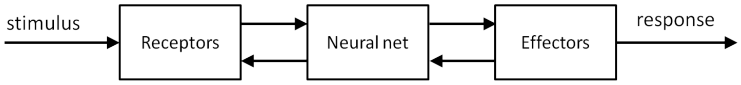


Figure 3.1: A schematic representation of the nervous system

human brain and artificial neural network, the neurons under stimulation send out electrical pulses (or spikes) to communicate with other neurons connected to itself using particular connections (synaptic connections). The engineering perceptron neuron model (see Fig. 3.2), proposed by Rosenblatt [34] in 1958, has the same generic characteristics as human ones: the NN is stimulated by input signals (x_n), and the stimulus is sent, using dedicated links (w_{jn}), to a mathematical function (or activation function f_j) which elaborates an output response (y_i). According to the perceptron neuron model in Fig. 3.2, the $j-th$ neuron is mathematically described by Eq. 3.1

$$y_j = f_j(v_j) = f_j\left(\sum_{i=1}^{n} w_{ji}x_i + b_j\right). \tag{3.1}$$

In order to highlight the working of the artificial neural network and the importance of the bias role, consider, for example, modeling a thermocouple [45]. The governing

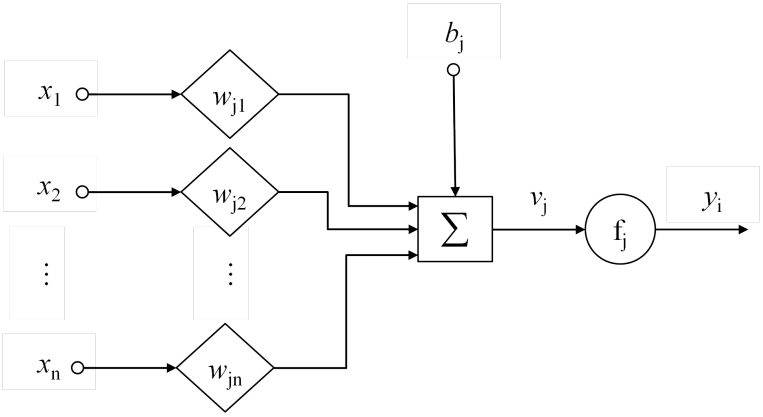


Figure 3.2: Perceptron neuron model with non-linear activation function f_j and bias b_j

equation can linearly be approximated as (3.2) within a certain range of temperatures for which the linear hypothesis is valid,

$$V_{out} = K_T \Delta T + b. \tag{3.2}$$

Using a neural network with only one neuron and a linear activation function ($f = 1$), the neural network approximation is

$$\hat{y} = w_{11} x_1 + b_1. \tag{3.3}$$

The importance of the bias, for any single neuron, is clear from Eq. 3.3 because it allows one to choose from among a parallel line set. Obviously, after the learning process, the closer the weight w_{11} is to K_T, the more the neural network will be accurate.

Having proved that neural networks can approximate continuous multi variable functions, a method, known as the *learning process*, is needed to optimize the free parameters of neural networks.

The key to success of ANNs is represented by

- the synaptic weights optimized using a learning algorithm;
- the type of activation functions.

These two topics will be discussed in the next sections.

Generally speaking, an artificial neural network is made up of several neurons organized in different layers, as depicted in Fig. 3.3. This particular neural network

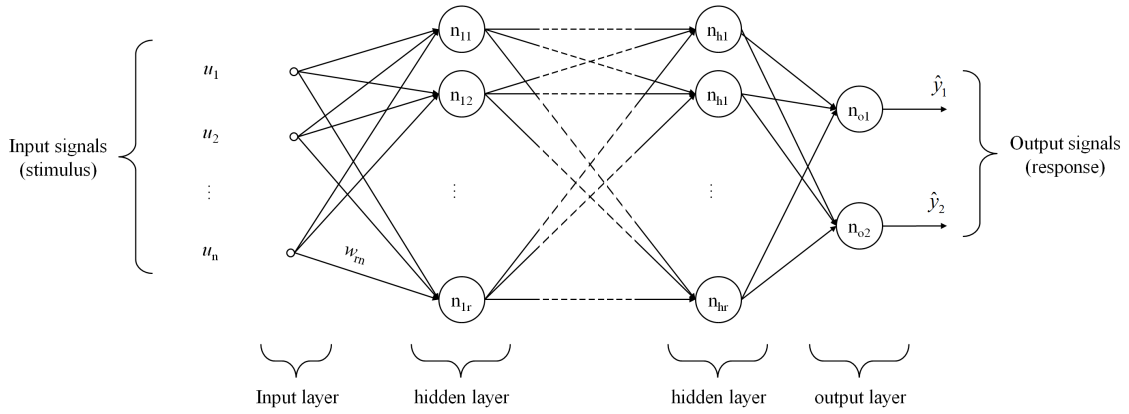


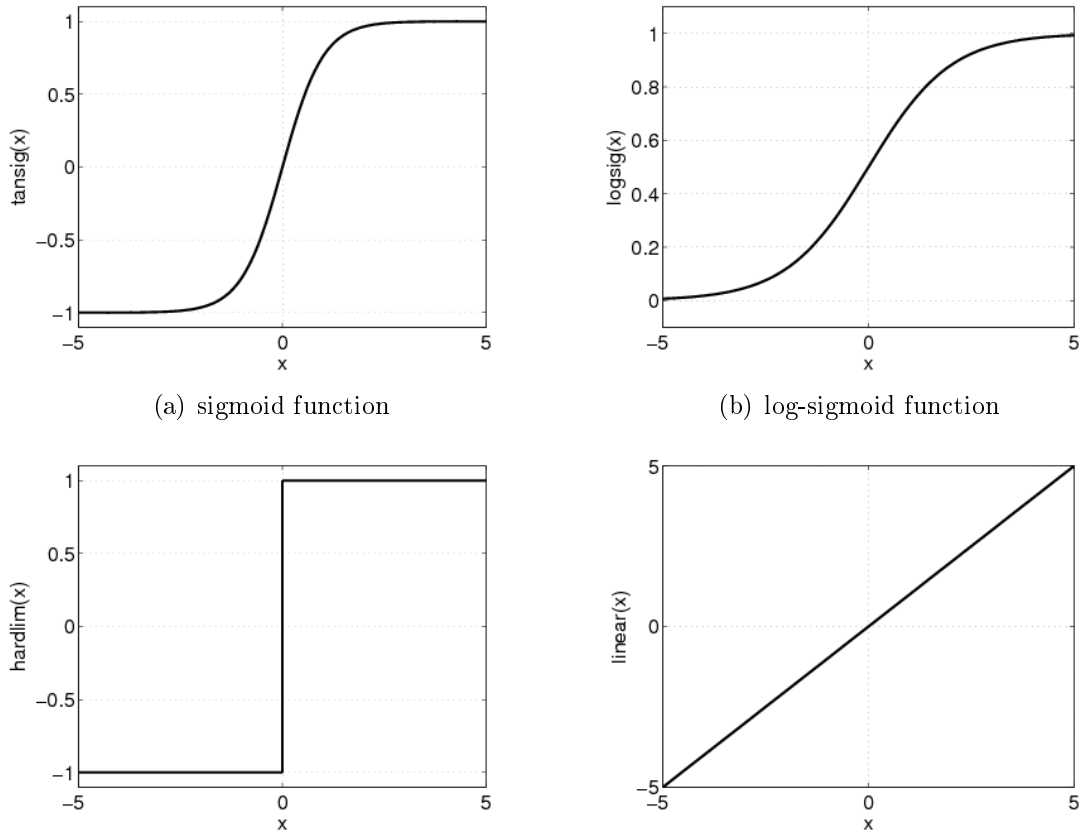
Figure 3.3: General architecture of a feedforward multilayer neural network

estimates two outputs, $\hat{y}_1$ and $\hat{y}_2$, and has n inputs in the input layer, r neurons in each of the h hidden layers and two neurons in the output layer. The mathematical model of Fig. 3.3 is very similar to the human nervous system depicted in Fig. 3.1. The sensory unit is the source node that constitutes the input layer, or input vector, while the hidden layers represent the neural network and the effectors are the computation node in the output layer. This kind of neural network is commonly known as a multilayer perceptron (MLP). Each neuron in Fig. 3.3, n_{hr}, in turn contains the scheme of Fig. 3.2, therefore the outputs are calculated as in Eq. 3.4

$$\hat{y}_{1,2} = f_{o1,2}\left(\sum_{l=1}^{r} f_{hl}\cdots\left(f_{1i}\left(\sum_{k=1}^{n} w_{ik}x_k + b_{1i}\right)\right)\right). \tag{3.4}$$

3.2.1 Activation Functions

Any kind of mathematical function that is defined in the input domain can be used as an activation function. Here, a list of the nonlinear activation functions that were considered in this work is given in Fig. 3.4. The linear function (Fig. 3.4(d)) is commonly used as an activation function for the output neurons in order to give, to neural networks, the ability to extrapolate beyond the training limits, which is not possible with a limited function. Instead, the linear function suffers from diverging output: the greater the stimulus, the greater the response. For this reason, the activation functions of inner neurons usually have limited functions, such as the sigmoid type, in order to prevent any diverging output of the inner neurons from



(a) sigmoid function

(b) log-sigmoid function

(c) hard-limiter function

(d) linear function

Figure 3.4: Examples of activation functions

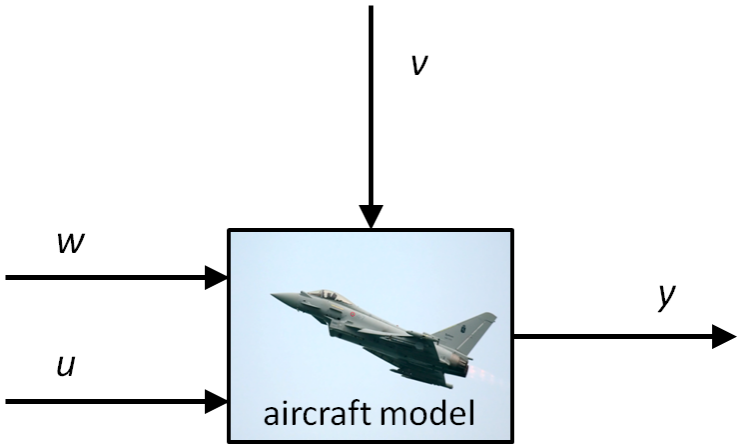


Figure 3.5: A system with input u, output y, measured disturbances w and unmeasured disturbances v

going to the output layer.

The hard limitier (Fig. 3.4(c)) is another limited function but it has the drawback of having non-continuous derivatives. The logarithmic sigmoid function (3.4(b)), positive defined, has been shown to be less suitable because it does not offer a good performance for the present application.

3.3 System Identification Methods

In order to study the dynamic behavior of real complex systems, such as aircraft, a mathematical model can be used instead of an actual test to reduce costs and time. The development of mathematical model starts from classical dynamic equations, which are approximated according to the level of accuracy required for the simulations, and then detailed with known parameters, such as aerodynamic coefficients from wind tunnel tests. The model is frozen when a comparison between the simulated and observed data from real system tests is used to tune some of the uncertain parameters related to the mathematical model. Consider the general aircraft system in Fig. 3.5. The main block is made up of several sub-blocks, each of which mathematically reproduces real sub systems, e.g. the engine block. The signals manipulated by an external user, such as the pilot commands, are called inputs, u, while the observable signals, as well as the vertical acceleration, are called outputs, y. The external disturbances that can be measured, such as the compressor rotating

velocity, or cannot be measured, such as the actual air turbulence, are also indicated. Real-life systems are quite complex to model with high accuracy. Model developers sometimes make use of other different techniques from conventional model based ones to model actual systems.

System identification methods are widely used when the complexity of the system and the processes involved are high, because in this way the model can be built even if the governing equations are unknown. The model is in fact only built using observed data from the actual system: the input and output patterns are recorded and subjected to data analysis to infer a model. One of the keys to success of system identification methods is the correct composition of the so-called *regression* vectors, which are used to predict the future output of a system. Here, a simple linear system, which satisfies (3.5), is described in order to introduce some quantities and notations that will be used in this work.

$$y(t)+a_1y(t-1)+\ldots+a_ny(t-n)=b_0u(t)+b_1u(t-1)+\ldots+b_mu(t-m). \quad (3.5)$$

The linear system of (3.5) has been chosen to represent the system in discrete time, because the observed data are always collected through sampling. The same system can be written in a deterministic way, see Eq. (3.6), where the output at time t is expressed as a function of the previous observations

$$y(t)=b_0u(t)+b_1u(t-1)+\ldots+b_mu(t-m)-a_1y(t-1)-\ldots-a_ny(t-n), \quad (3.6)$$

Eq. (3.6) can be rearranged in more compact notation form by introducing the vectors

$$y(t)=\underline{\phi}^T(t)\underline{\theta}, \quad (3.7)$$

where

$$\underline{\phi}(t)=\left[u(t),\ldots,u(t-m),-y(t-1),\ldots,-y(t-n)\right]^T, \quad (3.8)$$

is the regression vector and its components are the regressors, and

$$\underline{\theta}=\left[b_0,\ldots,b_m,a_1,\ldots,a_n\right]^T. \quad (3.9)$$

The model described in Eq. (3.5) calculates the output "regressing" or going back to the regression vector, $\underline{\phi}(t)$. It is also partly "Auto-Regressive" since the regression

vector $\underline{\phi}(t)$ contains old values of the variable that has to be calculated, $y(t)$. The model structure in Eq. (3.5) has the standard name of ARX-model: Auto-Regression with eXtra inputs (or eXogenous variables). Once the regression vector structure has been defined, i.e. how many old outputs and inputs have to be used, the ARX-model performance depends on the definition of vector $\underline{\theta}$. The ARX-model often cannot be used in real life applications because the exact old outputs, y, are not available, so the ARX-model can be substituted with the Output-Error model (OE) according to which the estimated outputs are used in the regression vector as regressors.

$$\underline{\phi}(t) = [u(t), \ldots, u(t-m), -\hat{y}(t-1), \ldots, -\hat{y}(t-n)]^T, \tag{3.10}$$

The regression vector (3.10) of the OE-model is used in this work in order to avoid the need of knowing the exact outputs. Moreover, the use of outputs in the regression vector is rather tricky because some errors are inserted into the input layer at each time step from the old estimated output. To overcome this problem, the estimated outputs were not considered in the regression vector in this work.

3.4 Multilayer Perceptron

The single perceptron model was introduced by Rosenblatt [34], and it remains the simplest form of neural network. As previously mentioned, the neural network depicted in Fig. 3.3 represents a common multilayer perceptron, which has been successfully applied to solve several problems by means of training using algorithms that belong that belong to the error correction learning process, which is characterized by two ways of propagating through the layers, the forward pass and the backward pass. In the forward pass, the input vector is applied to sensory nodes of the input layers, and its effects propagates, layer by layer, to the output nodes. Here, a set of outputs were calculated and the error of estimation was calculated comparing the NN response with the desired target. During the backward pass, the error flows from the output node to the first hidden layer (see Fig. 3.6). The synaptic weights and bias levels are adjusteds, by means of error correction rules, in order to minimize the distance between the neural network prediction and the desired target. However, whatever the learning algorithm is, a common learning loop, depicted in Fig. 3.7, is

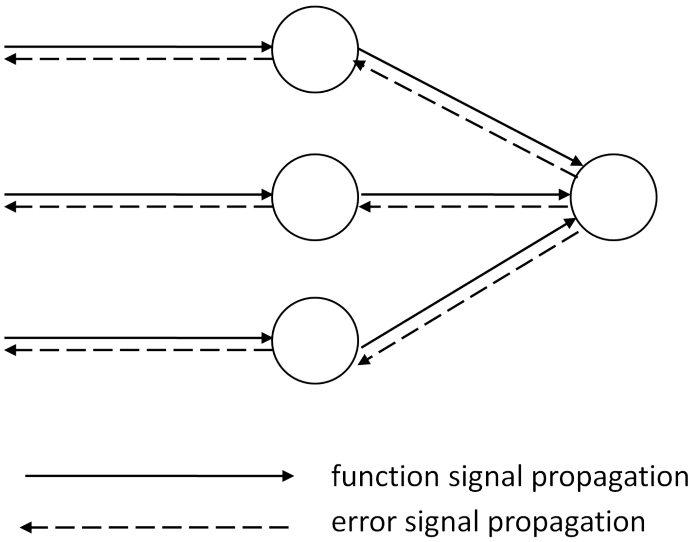


Figure 3.6: Illustration of the two propagation ways for signals and errors in the error correction algorithm

followed. In the first step, the synaptic weights and bias levels (neural network free parameters) are randomly initialized, then the neural network response is calculated. The error in NN estimation is calculated with respect to the desired target, d. The error is used to update the free parameters according to the chosen learning, or training, algorithm until the convergence criterion is satisfied.

3.4.1 Neural Network as a System Identification Method

The purpose of this section is to introduce the connecting bridge between system identification methods and artificial neural networks. Sometimes, neural networks are referred to as *black box* system identification techniques for nonlinear dynamic systems, because of the non physical meaning of the synaptic weight. Neural network design is subject to the same rules as the system identification presented in section 3.3. The neural network structure can in fact be chosen through the following two steps:

- selection of the inputs to the network in order to define the regression vector
- selection of the neural network architecture in order to define vector $\underline{\theta}$.

The first step is equivalent to selecting a model, such as the ARX or OE, or even the NNARX and NNOE model if referring to an artificial neural network. Other model

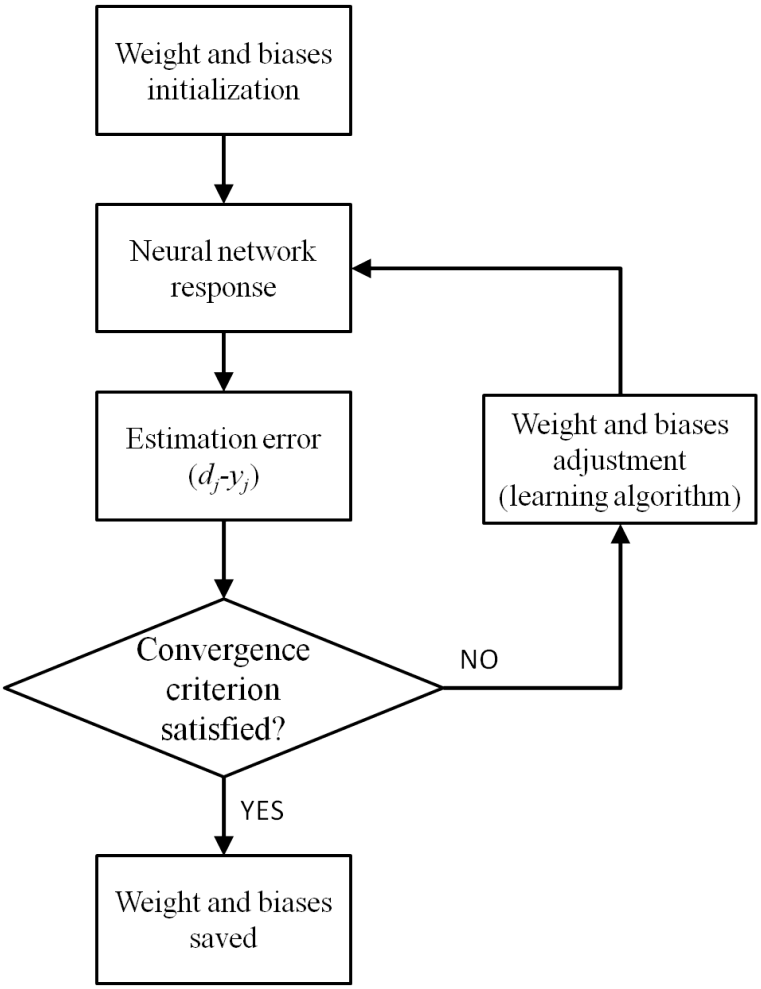
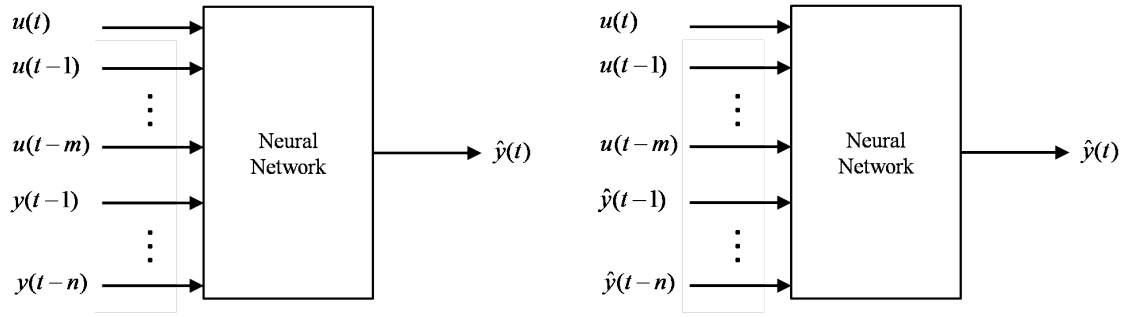


Figure 3.7: Flow-chart of the standard learning process



(a) The NNARX model structure (b) The NNOE model structure

Figure 3.8: Neural network models

structures taht were not considered in this work can be found in [46, 47, 48, 49, 50]. When defining a neural network three main steps are required

1. the choice of inputs to be used as stimulus of to the network from among the available signals;
2. the choice of how many old, or past, inputs should be used in the input vectors
3. the choice of an activation function

Consider, for example, the SISO neural networks shown in Fig. **??**. Since there is only one input, u, the choice of input signal is forced, but in general a detailed analysis of the system at hand is required to establish the necessary inputs. Here, the system analysis refers to a mathematical description of the real-world system to be modeled with the aim of highlighting the independent variables; an example is explained in detail in the section 4.1. However, the number of old inputs and outputs is chosen by the user according to her/his previous experience or on the basis of a real-world system analysis. Generally speaking, each input, and output, may have its own number of past inputs, even though this practice is not very common and often refers to neural network architecture optimization that goes beyond the goal of this work. Once the choice of inputs has been made, the regression vector is defined as in (3.8) for the NNARX model and (3.10) for the NNOE model. The neural network architecture, i.e. the numbers of neurons and hidden layers, is designed, and successively optimized, in order to satisfy the network performance specifications and the best generalization behavior. Performance and generalization will be defined

in the next sections. Once the architecture has been frozen, vector $\underline{\theta}$, which is made up of neural network free parameters, is also defined. Overall, if the neural network input vector is defined, the regression vector, $\underline{\phi}$, is also consequently defined, whereas if the neural network architecture is defined, the vector $\underline{\theta}$ is fulfilled with synaptic weights and biases. The neural network prediction can be written as $\hat{y}(t) = \underline{\phi}^T(t)\underline{\theta}$, or using more conventional neural network notations as

$$\hat{y}(t) = \underline{\phi}^T(t)\underline{w}(t). \tag{3.11}$$

Therefore, the neural network problem can be considered as a system identification problem and all the known optimization techniques suitable for system identification can be used to train the neural networks with the aim of reducing the distance between the NN prediction and the desired output.

3.5 Neural Network Training

In 1970, Mendel and McClaren [51] defined the learning process as "the process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place". The training process is thus a set of well defined rules that allows the neural network to learn from an input-output data pattern and hence to adapt itself to its environment. In other words, during the learning process the synaptic weights are changed with the aim of improving the neural network performance, which is usually referred to as the error of NN the prediction of given continuous multivariate non-linear functions, such as complex real-world systems.

Consider, for the safe of simplicity, a neural network with only the j-th neuron depicted in Fig. 3.2 in the output layer. The neuron is driven by an input vector, $\underline{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, which is applied upstream in the input layer or produced by previous hidden layers. The error of the neural network prediction is obtained comparing the NN output signal, y_j, and the desired target d_j

$$e_j(t) = d_j(t) - y_j(t). \tag{3.12}$$

The final goal of the neural network training process is to minimize the output error of (3.12), or to match other requirements specified by the user. The algorithm used to minimize (3.12) defines the learning method.

Several learning methods exist [48, 46, 47]: supervised, reinforcement and unsupervised. In supervised learning, the learning process incorporates an external teacher and/or performance information through a training set (input-output) of desirable network behavior; indeed both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system in order to optimize the synaptic weights and biases.

It has been assumed that the desired target outputs are known for each input pattern. However, in some situations, less detailed information is available, for example, whether the output is right or wrong. Under this hypothesis, supervised learning is not feasible and a reinforcement learning method must be used. Sometimes the reinforcement learning technique is seen as a particular form of supervised learning, because the network, obtains some feedback from the surrounding environment.

There is no teacher in unsupervised learning. We still consider a network with both inputs and outputs, but there is no feedback from the environment to say what the desired output is or whether it is correct. The network itself must discover patterns, features, correlations or categories in the input data and code them in the output. The units and connections must thus display some degree of self-organization capabilitiy. Currently, in fact, this learning method is limited to networks known as self-organizing maps and are not in widespread use.

Overall, the best learning method for this work is the supervised method. Within supervised learning. Several algorithms exist, here gradient methods will be considered in particular, and first order and second order learning rules will be considered. As shown in the next two sections, two classical algorithms were considered for the training stages in this work: back propagation (BP) and Levenberg-Marquardt (LM) algorithms. The BP method is one of the most frequently used in the past training techniques, conversely LM is a second order training method and it was proved to be faster than BP.

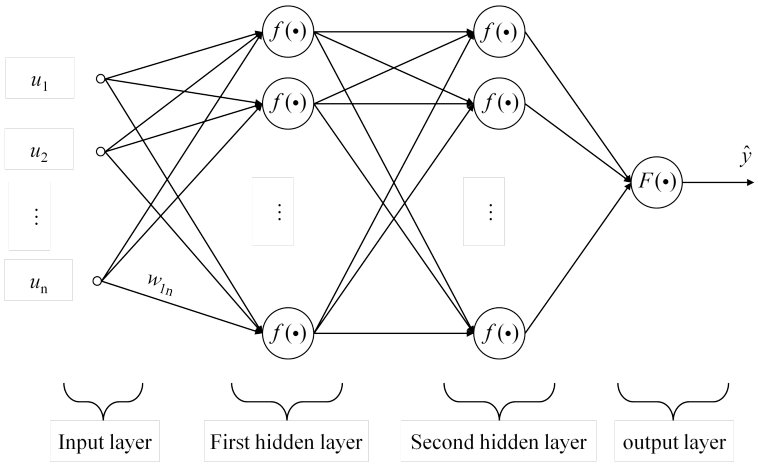


Figure 3.9: Multi layer perceptron neural network with two hidden layers with the same activation function and one output layer

3.5.1 Error Back Propagation Algorithm

The backward propagation of error method was first published in 1969 by Bryson and Ho [52]. A few years later, in the early 1970's, several other authors independetly developed the back-propagation technique. In 1974, Werbos [53] applied it to behavioural sciences, and only in 1986 was it applied to the training of multi-layer networks and called *back-propagation* by Rumelhart, Hinton and Williams [54].
Consider, for the sake of simplicity, a neural network with two hidden layers with the same activation function, f, and one neuron in the output layer, as illustrated in Fig. 3.9. The error of NN estimation, as defined in (3.12), can be calculated for each time step at the output node, and the error energy for the j-th output neuron at time step t can be written as

$$\mathscr{E}(t) = \frac{1}{2}\sum_{j\in C} e_j^2(t) = \frac{1}{2}\sum_{j\in C}\left(d_j(t) - y_j(t)\right)^2, \tag{3.13}$$

where C is the output signal set. If the network has only one output, $C = 1$ and (3.13) will simplify to $\mathscr{E}(t) = \frac{1}{2}e^2(t)$. Equation (3.13) represents the cost function of the learning performance that has to be minimized by adjusting the free parameters of the neural network. The back propagation algorithm applies a correction, Δw_{ji}, to the synaptic weight, w_{ji}, which is proportional to the gradient $\partial\,\mathscr{E}(t)/\partial w_{ji}$. According to the chain rule, the gradient $\partial\,\mathscr{E}(t)/\partial w_{ji}$ can be written as

$$\frac{\partial\,\mathscr{E}(t)}{\partial w_{ji}(t)} = \frac{\partial\,\mathscr{E}(t)}{\partial e_j(t)}\frac{\partial e_j(t)}{\partial y_j(t)}\frac{\partial y_j(t)}{\partial v_j(t)}\frac{\partial y_j(t)}{\partial w_{ji}(t)}. \tag{3.14}$$

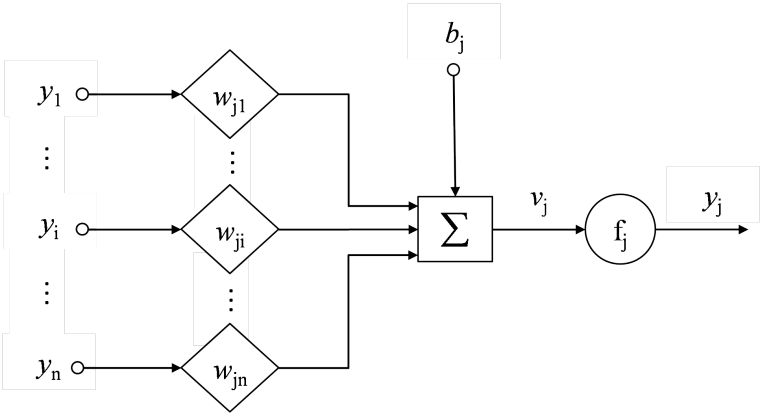


Figure 3.10: Signal flow scheme of the j-th output neuron

Considering the neuron in Fig. 3.2 to be an output neuron, the i-th input, x_i, represents the output of the previous neurons, and, for conformity of notation, it will be indicated later in this work as y_i (see Fig. 3.10). Eq. (3.1) may be rewritten as

$$y_j = f_j(v_j) = f_j\left(\sum_{i=1}^{n} w_{ji}y_i + b_j\right) = f_j\left(\sum_{i=0}^{n} w_{ji}y_i\right), \tag{3.15}$$

where the synaptic weight w_{j0} (corresponding to the fixed input $y_0 = 1$) equals the bias b_j. By differentiating Eq. (3.15) with respect to $v_j(t)$ and $w_{ji}(t)$ separately, the following equations can be obtained

$$\frac{\partial y_j(t)}{\partial v_j(t)} = f_j'(v_j(t)) \tag{3.16}$$

and

$$\frac{\partial v_j(t)}{\partial w_{ji}} = y_i(t) \tag{3.17}$$

By differentiating both sides of (3.12) with respect to $y_j(t)$, and (3.13) with respect to $e_j(t)$, we obtain

$$\frac{\partial e_j(t)}{\partial y_j(t)} = -1 \tag{3.18}$$

and

$$\frac{\partial\, \mathscr{E}(t)}{\partial e_j(t)} = \sum_{j\in C} e_j(t). \tag{3.19}$$

Therefore, the use of equations (3.16) to (3.19) in (3.14), yields

$$\frac{\partial\, \mathscr{E}(t)}{\partial w_{ji}(t)} = -y_i(t)\sum_{j\in C} e_j(t)f_j'\left(v_j(t)\right) = -y_i(t)\delta_j(t), \tag{3.20}$$

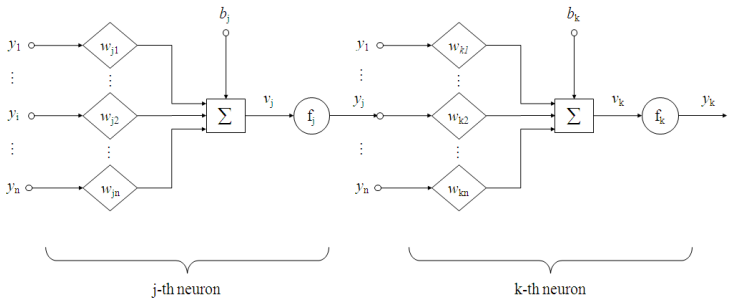


Figure 3.11: Signal flow scheme of the j-th hidden neuron

where $\delta_j(t)$ is commonly defined as the local gradient with the following expression for the output layer

$$\delta_j(t) = -\frac{\partial\, \mathscr{E}(t)}{\partial e_j(t)} \frac{\partial e_j(t)}{\partial y_j(t)} \frac{\partial y_j(t)}{\partial v_j(t)} = -\frac{\partial\, \mathscr{E}(t)}{\partial y_j(t)} \frac{\partial y_j(t)}{\partial v_j(t)} = \sum_{j \in C} e_j(t) f_j'\left(v_j(t)\right). \quad (3.21)$$

The correction to the synaptic weight $w_{ji}(t)$ is established using the delta rule

$$\Delta w_{ji}(t) = -\eta \frac{\partial\, \mathscr{E}(t)}{\partial w_{ji}(t)} = \eta \delta_j(t) y_i(t), \quad (3.22)$$

where η is defined as the learning rate. Therefore, the updated weight is

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) = w_{ji}(t) - \eta \frac{\partial\, \mathscr{E}(t)}{\partial w_{ji}(t)} = w_{ji}(t) + \eta \delta_j(t) y_i(t). \quad (3.23)$$

As shown by several authors [46, 48], there is no optimum learning rate, but according to the particular problem, there is a specific η that assures fast and stable convergence. Some algorithms are provided with a learning algorithm rate that varies according to the local gradient, or other parameters, to speed up the convergence [55].

If the j-th neuron is part of a hidden layer, the error flows backward through the output layer before getting to a hidden layer. In order to highlight this process, consider Fig. 3.11. In analogy with the previous j-th output neuron treatment, the chain rule is used to find a useful expression of the gradient $\partial\, \mathscr{E}(t)/\partial w_{ji}(t)$.

For hidden neurons, the local gradient has the same expression as (3.21), but, this time, the index j represents the hidden neurons. In (3.13), index j must be substituted with k, to agree with the subscripts in Fig. 3.11, in order to obtain the new expression of the neural network error

$$\mathscr{E}(t) = \frac{1}{2} \sum_{j \in C} e_j^2(t) = \frac{1}{2} \sum_{k \in C} \left(d_k(t) - y_k(t)\right)^2. \quad (3.24)$$

By differentiating (3.24) with respect to $y_j(t)$, it yelds

$$\frac{\partial\, \mathscr{E}(t)}{\partial y_j(t)} = \sum_{k\in C} e_k(t)\frac{\partial e_k(t)}{\partial y_j(t)} = \sum_{k\in C} e_k(t)\frac{\partial e_k(t)}{\partial v_k(t)}\frac{\partial v_k(t)}{\partial y_j(t)}. \tag{3.25}$$

Considering Fig. 3.10, it is easy to obtain

$$e_k(t) = d_k(t) - y_k(t) = d_k(t) - f_k(v_k(t)). \tag{3.26}$$

Hence, by differentiating with respect to $v_k(t)$, we obtain

$$\frac{\partial e_k(t)}{\partial y_j(t)} = f_k'\left(v_k(t)\right). \tag{3.27}$$

Equation (3.15) can be rewritten for the k-th neuron and differentiated, with respect to $y_j(t)$, in order to obtain the following expression

$$\frac{\partial v_k(t)}{\partial y_j(t)} = w_{kj}(t). \tag{3.28}$$

Using (3.28), (3.27) and (3.25), it yields

$$\frac{\partial\, \mathscr{E}(t)}{\partial y_j(t)} = -\sum_{k\in C} e_k(t) f_k'(v_k(t)) w_{kj}(t) = -\sum_{k\in C} \delta_k(t) w_{kj}(t) \tag{3.29}$$

Finally, by using (3.29) in (3.14), the back propagation formula for the hidden neuron is obtained as follows

$$\delta_j(t) = f_j'\left(v_j(t)\right) \sum_{k\in C} \delta_k(t) w_{kj}(t), \tag{3.30}$$

where the expression for δ_k is (3.21), in which j is substituted by k. At the end, the adjusting weight for a hidden neuron may be expressed as

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) = w_{ji}(t) - \eta \frac{\partial\, \mathscr{E}(t)}{\partial y_j(t)}\frac{\partial y_j(t)}{\partial v_j(t)}\frac{\partial v_j(t)}{\partial w_{ji}(t)} = w_{ji}(t) + \eta \delta_j(t) y_i(t), \tag{3.31}$$

This is one of the oldest and most frequently used weight adjustment algorithms. The back-propagation algorithm (BP) presented here can only be used for *on-line* training, or, in other words, for step by step training. The BP algorithm can also be used in batch mode, as described in the next section, in order to be used as an *off-line* training type.

3.5.2 Batch Error Back-Propagation Algorithm

As can be noted in (3.30) and (3.22), a correction is applied for each time step, or iteration, of the training data set that is characteristic of the *on-line* training strategy. In order to use the error back-propagation technique in *off-line* mode, the whole available input-output pattern is used in a single training run. In this way, the network synaptic weights are updated after that the entire training data set has been inferred to the neural network. In order to consider the whole corresponding input output set, 3.12 is rewritten for batch BP (BBP) as

$$\mathscr{E}_{av} = \frac{1}{N}\sum_{t=1}^{N} \mathscr{E}(t) = \frac{1}{2N}\sum_{t=1}^{N}\sum_{j\in C} e_j^2(t) = \frac{1}{2N}\sum_{t=1}^{N}\sum_{j\in C} \left(d_j(t) - y_j(t)\right), \qquad (3.32)$$

and a very similar procedure to that presented in the previous section can be adopted to obtain the updating weight. In order to accelerate the convergence of the BP algorithm, a second order training algorithm will be presented in the next section.

3.5.3 Descent Methods

Since the Levenberg-Marquardt (LM) algorithm belongs to the Newton method category, which consists of gradient-based methods, a brief description of these methods is presented here: first a digression on gradient and Newton's methods will be presented, and then the LM algorithm will be discussed.
As for the back propagation algorithm, the focus is on minimizing the error function $\mathscr{E}$. Moreover, in order to simplify the treatment, the *on-line* training method will be presented since very few modifications are required to obtain batch, or *off-line*, versions.

Gradient-based Methods

As shown for the BP algorithm, the main objective is to minimize the performance function, $\mathscr{E}$ of (3.12), defined on an n-dimensional input space $\underline{w} \in \mathbb{R}^R$, where the synaptic weights and biases are arranged in a one dimensional vector, $\underline{w} =$

$[w_1, \ldots, w_R]^T$, where R is the number of all the free parameters of the neural network. The gradient expression on the left hand side of 3.14 can be expressed as a function of all the weights

$$\underline{g}(t) = \frac{\partial \mathscr{E}(t)}{\partial \underline{w}(t)} = \left[\frac{\partial \mathscr{E}(t)}{\partial w_1(t)}, \ldots, \frac{\partial \mathscr{E}(t)}{\partial w_R(t)}\right]^T. \tag{3.33}$$

According to gradient descent methods, the update weights, $\underline{w}$, are calculated as

$$\underline{w}(t+1) = \underline{w}(t) - \eta \underline{\underline{G}}\underline{g}(t) = \underline{w}(t) - \eta \frac{\partial \mathscr{E}(t)}{\partial \underline{w}(t)}, \tag{3.34}$$

where $\underline{\underline{G}}$ is a certain positive definite matrix.

Steepest Descent Method

The steepest descent method is one of the oldest technique used to minimize multivariate functions. For this method, the matrix $\underline{\underline{G}}$ is defined as the identity matrix, $\underline{\underline{I}}$, hence the (ErrorgradientLM) becomes the well known

$$\underline{w}(t+1) = \underline{w}(t) - \eta \underline{g}(t). \tag{3.35}$$

The negative search direction, $-\underline{g}$, means this method follows the *local steepest descent downhill* direction, which implies the algorithm is effected to a great extent by the initial conditions and easily *falls* in local minima, without globally minimizing the given multivariate function $\mathscr{E}$.
Comparing (3.35) and (3.23), it can be noted that the equations are rather familiar. The steepest descent method only differs from the back-propagation method as far as the formal analytical expressions are concerned, but they are essentially driven by the same principles: the updating weights depend on the gradient of function $\mathscr{E}$. The BP algorithm is in fact often identified with the steepest descent method.

Newton's Method

The descent direction for Newton's method is determined using the second order derivatives of the objective function, if they are available. Apart from the mathematical

concerns about the existence of derivatives, the performance function $\mathscr{E}$ can be unfolded by means of a Taylor series and taken at second-order approximation

$$\begin{aligned}\mathscr{E}(t+1) \approx\ & \mathscr{E}(t) \\ & + \underline{g}^T [\ \underline{w}(t+1) - \underline{w}(t-1)\ ] \\ & + \frac{1}{2} [\ \underline{w}(t+1) - \underline{w}(t-1)\ ]^T \underline{\underline{H}} [\ \underline{w}(t+1) - \underline{w}(t-1)\ ],\end{aligned} \tag{3.36}$$

considering that higher order terms are omitted in the assumption that $[\ \underline{w}(t+1) - \underline{w}(t-1)\ ]$ is sufficiently small and $\underline{\underline{H}}(t)$ is the Hessian matrix

$$\underline{\underline{H}}(t) = \begin{bmatrix} \dfrac{\partial^2\ \mathscr{E}(t)}{\partial^2 w_1(t)} & \cdots & \dfrac{\partial^2\ \mathscr{E}(t)}{\partial w_1(t) \partial w_R(t)} \\ \\ \cdots & \cdots & \cdots \\ \\ \dfrac{\partial^2\ \mathscr{E}(t)}{\partial w_R(t) \partial w_1(t)} & \cdots & \dfrac{\partial^2\ \mathscr{E}(t)}{\partial^2 w_R(t)} \end{bmatrix}. \tag{3.37}$$

Since (3.36) is a quadratic function of $\underline{w}(t)$, its minimum point ($\underline{w}_{MIN}$) can easily be obtained by differentiating (3.36) and making it equal to zero, and then solving the following set of linear equations

$$0 = \underline{g}(t) + \underline{\underline{H}}(t) [\ \underline{w}_{MIN} - \underline{w}(t+1)\ ]. \tag{3.38}$$

If the inverse matrix of $\underline{\underline{\mathrm{H}}}$ exists, Newton's method is obtained as

$$\underline{w}_{MIN} = \underline{w}(t) - \underline{\underline{H}}(t)^{-1} \underline{g}(t). \tag{3.39}$$

It is clear that if the error function $\mathscr{E}$ is not quadratic, $[\ \underline{w}_{MIN} - \underline{w}(t+1)\ ]$ only represents a first step towards the minimum point. Therefore, several steps are needed to achieve the minimum point, using an iterative scheme and updating new weights as

$$\underline{w}(t+1) = \underline{w}(t) - \underline{\underline{H}}(t)^{-1} \underline{g}(t), \tag{3.40}$$

which is the general expression of (3.34), where $\underline{\underline{G}} = -\underline{\underline{H}}^{-1}$ and $\eta = 1$.

3.5.4 Levenberg-Marquardt Algorithm

If the inverse Hessian matrix exists but is not positive definite, Newton's method, as described before, can lead to a local maximum, or a saddle point [49]. To overcome this issue, Levenberg [56] and Marquardt [57] proposed altering the Hessian matrix with a positive definite matrix $\underline{\underline{P}}$ to make $\underline{\underline{H}}$ positive definite in least square problems. Later, Goldfeld et al. [58] applied this methodology to Newton's method, considering $\underline{\underline{P}} = \lambda \underline{\underline{I}}$. Therefore, the updated synaptic weight (3.40) can be re-written according to Levenberg-Marquardt as

$$\underline{w}(t+1) = \underline{w}(t) - \left(\underline{\underline{H}}(t) + \lambda \underline{\underline{I}}\right)^{-1} \underline{g}(t), \tag{3.41}$$

where $\underline{\underline{I}}$ is the identity matrix and λ is a certain non negative value. The LM method transits smoothly between Newton's method, as λ approaches 0, and the steepest descent method as λ grows infinitely.

In this work, λ was chosen according to variation of performance index for each training iteration (t). The steepest descent method is utilized at a large distance from the minimum of the considered function, to provide steady and convergent progress toward the solution. As the solution approaches the minimum, λ is adaptively decreased, the Levenberg-Marquardt method approaches Newton's method, and the solution usually converges rapidly to the local minima.

3.5.5 Validation

Once the neural network has been designed and trained, it needs to be validated with several test cases. The test data set must be different from the training one in order to establish whether the neural network is able to predict any input-output mapping of the system at hand within the training boundaries. The training loop in Fig. 3.7 can be enhanced with the validation stage described in Fig. 3.12. According to [48], a neural network is said to *generalize* well when the input-output patterns predicted by the network are acceptable for a test data contained within the training boundaries but that have never been used before. However, when a neural network is trained using too many, or repeated input-output examples, the network

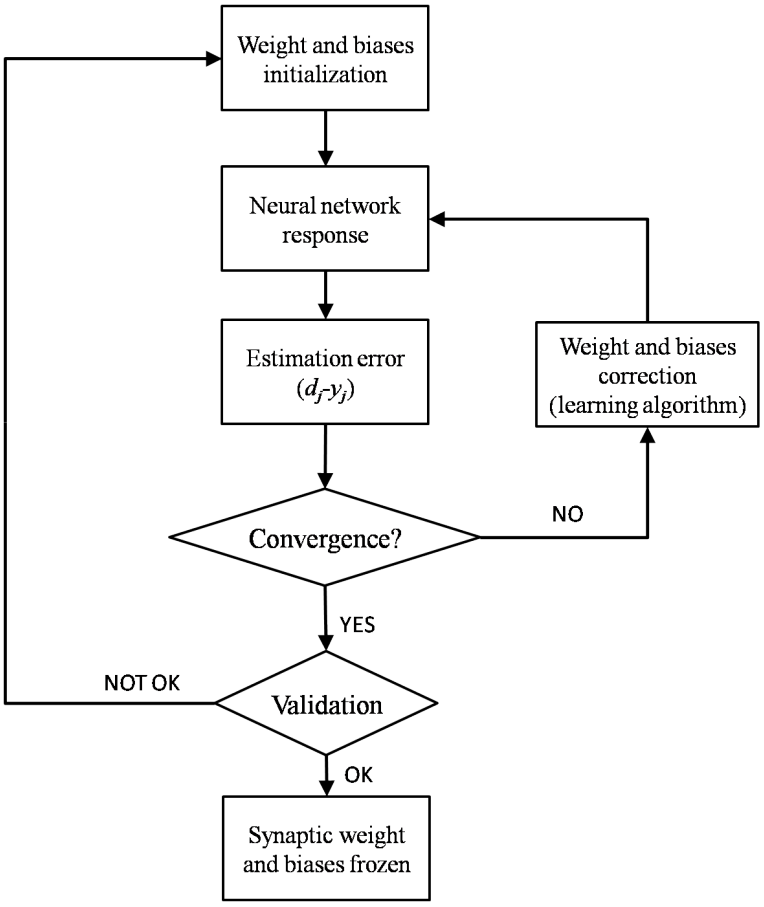


Figure 3.12: Flow-chart of the neural network training and validation process

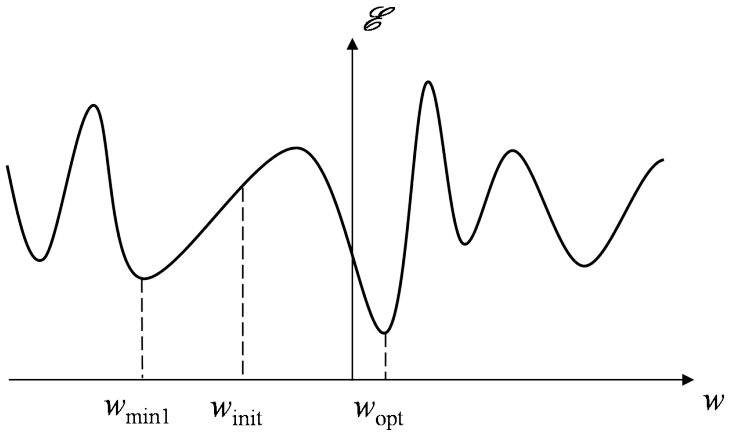


Figure 3.13: Non-linear curve-fitting problem with several local minima, highlighting the initial synaptic weight, w_{init}, a local minimum point, w_{min_1}, and the absolute minima, w_{opt}.

may memorize the training data, just like a human being. Such a phenomenon is referred to as *overfitting* or *overtraining*, and, in this case, the network looses the ability to generalize. Since, this deficiency is stored inside the synaptic weight, several technique have been developed such as network pruning techniques that will be discussed later on to adjust the weights in order to make the neural network more general.

Local minima is another issue that can be encountered when training a neural network. Consider, for example, a single layer neural network with one hidden neuron, a fixed bias and only one free synaptic parameter, w, as in Fig. 3.13. Therefore, as previously mentioned, the training can be seen as a non-linear curve-fitting problem, where the neural network error prediction, as defined in (3.13), depends on the value of the free parameter, w. Starting from the first weight attempt, w_{init}, any deterministic training algorithms, such as BP or LM algorithms, will always find the w_{min_1} as the best weight to minimize the error, or optimize the neural network performance. This example can be translated into multi-dimensional space and the error profile becomes a complex hyper surface which depends on the NN free parameters, synaptic weights and biases. The neural network training will often end up in local minima points of the hyper surface.

Many non-deterministic optimizing algorithms were introduced to overcome the local minima problem. GA and swarm are only two of the most famous optimizing

techniques. Since these methods require high computational resources and long times, due to the high number of independent variables, i.e. neural network free parameters, they were not used in this work because several re-initializations of synaptic weights were found to be suitable to avoid local minima and to be used simultaneously on parallel CPUs. At least ten re-initializations were considered adequate for the NN application of this work to choose the best NN training. Therefore, ten training processes were carried out at the same time of the single off-line training and then compared to choose the best trained neural network, considering the maximum error of the NN prediction using the test data set.
The method used in this work to overcome overfitting issues will be discussed later on.

3.5.6 Generalization - Network Growing and Pruning

Organisms generalize if they respond appropriately to stimuli and situations they have never experienced before [59]. As an organism's nervous system, a neural network should also be capable of generalizing, i.e., of generating the appropriate outputs in response to inputs that are not part of their training experience. A review of generalization methods for neural networks was made by Arbib in [60]. These methods share the final common goal of minimizing the errors of NN prediction on the training set and on the test set; a practical example is reported in the last section of the present chapter. A result of a generalization procedure is that the neural network will have the best performance with the minimum number of neurons. This is important because small neural network architectures are more suitable for use with limited computational resources and in short times. Moreover, adequate sized neural networks are less likely to suffer from learning noise from the training data.
All the aforementioned benefits can be achieved starting with the network *pruning* or *growing* techniques used in this work.
As far as the growing method is concerned, a small network is designed and a new neuron, or a new layer, is added to the network until the performance requirements are met.

The network pruning approach works in the opposite way to network growing. Starting from a large network for the problem at hand, some weights are weakened or deleted in a selective and orderly fashion, as can be seen in [48, 50].
These two methods suffer from the fact that anytime the neural network is modified, such as when weights are deleted or neurons are added, the network must be re-trained, while considering all the issues presented in the previous section. Even though the generalization techniques are very useful to make a neural network as general as possible, they are really time and hardware resource consuming.

3.6 NN Application: β Estimation Using Flush Ports

A simple FADS, made up of two static flush ports, was considered in order to calculate the angle of sideslip, β, for a fixed free stream velocity and null angle of attack. The aim of this example was to define some neural network procedures that can be used in the next applications of the present work.
The fuselage considered for this activity is inspired by the Sky-Y, Alenia Aermacchi unmanned aircraft. The aerodynamic analysis was carried out using CFD in order to correlate the angle of sideslip of fuselage and the static pressure sensed by several couple of virtual flush ports, positioned on the fuselage skin, as depicted in Fig. 3.14. The CFD simulations were performed using the commercial CFD code STAR-CCM+

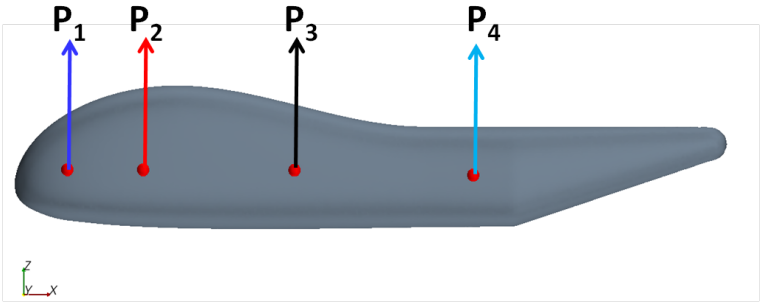
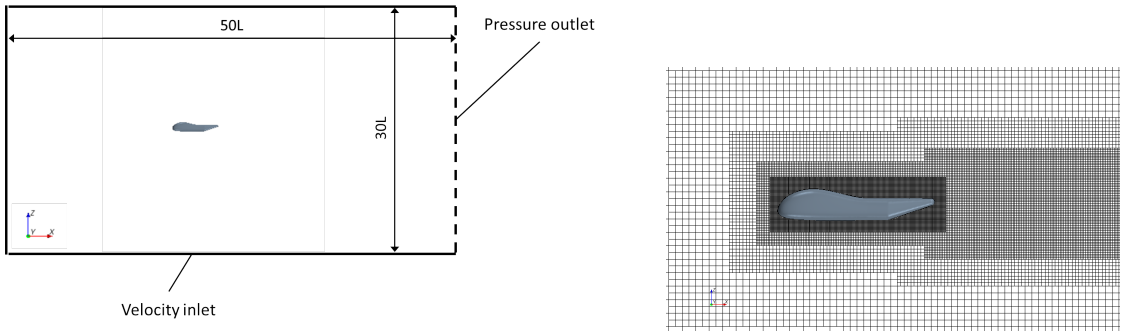


Figure 3.14: Virtual flush static ports on the left side of the fuselage. The probes are duplicated symmetrically on the other side of the fuselage

available at Alenia Aermacchi. The results of the CFD analysis were used to collect an input-output pattern for neural network training and testing in order to define all the procedures needed to correctly design a neural network exploiting this simple

test case.

More information about STAR-CCM+ can be found in [61]; just a few details are reported here. STAR-CCM+ is a three dimensional finite volume Navier-Stokes



(a) Boundary conditions (b) Close mesh view

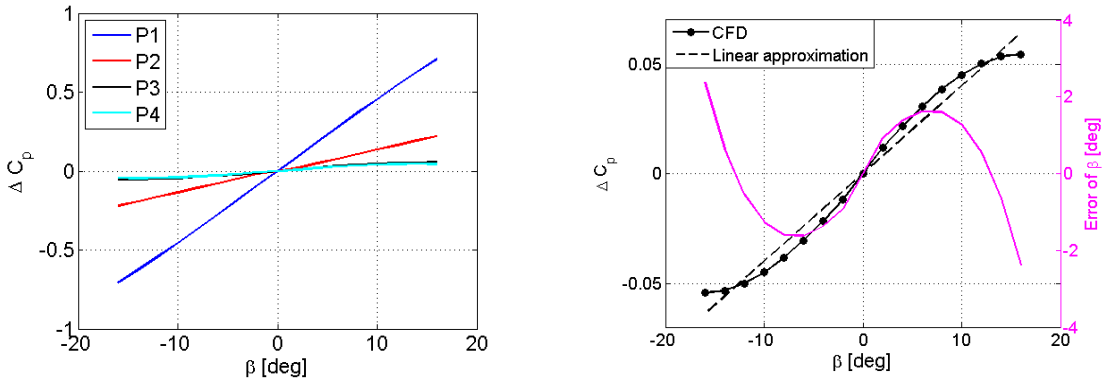
Figure 3.15: Computational domain, where L represents the fuselage length.

equation solver. The internal automatic mesher is used to generate the computational grid. The computational domain is presented in Fig. 3.15, where the inlet velocity condition, $V_\infty = 100\ kts$, is used for front and side walls, while the pressure outlet is used for the downstream wall. The β can be calculated as a function of the sole sensed static pressure on two opposite sides and of the dynamic pressure

$$\beta = f(q_c,\ p_{s,R},\ p_{s,L}) = f(C_{p_{s,R}},\ C_{p_{s,L}}) = f_1(\Delta C_p), \tag{3.42}$$

where $\Delta C_p = \frac{p_{s,R} - p_{s,L}}{q_{c_\infty}}$ is the normalized differential static pressure. ΔC_p, sensed at each of the four considered positions, is depicted in Fig. 3.16(a).

Only position $P3$ is considered with this application and the corresponding differential pressure coefficient, ΔC_p, is plotted in Fig. 3.16(b); the linear approximation, as described in (1.1), and the corresponding error of measured β are also plotted. Generally speaking, two main matters should be taken into account when deciding on the position of the static ports in order to measure the angle of sideslip. First, the ratio $\frac{\Delta C_p}{\Delta \beta}$; second, the linearity of the relationship between β and ΔC_p, if a linear approximation, such as (1.1), is considered to be used. The aforementioned ratio is extremely important to obtain a certain β resolution. A couple of static ports in position $P3$ would in fact only lead to a β resolution of $\approx 0.5\ deg$ considering state-of-the-art differential pressure sensors [62], whereas the resolution could be about 0.05 deg and 0.1 deg, respectively, in $P1$ and $P2$ using the same sensor.



(a) ΔC_p at several positions

(b) ΔC_p at position $P3$, its linear approximation and the corresponding β-error

Figure 3.16: Relationship between β and ΔC_p calculated using CFD

Therefore, since the more advanced the position is the higher the ratio of $\frac{\Delta C_p}{\Delta \beta}$, as depicted in Fig. 3.16(a), positions near the fuselage nose allow sensors to measure β with higher resolution, considering the same pressure sensor accuracy.

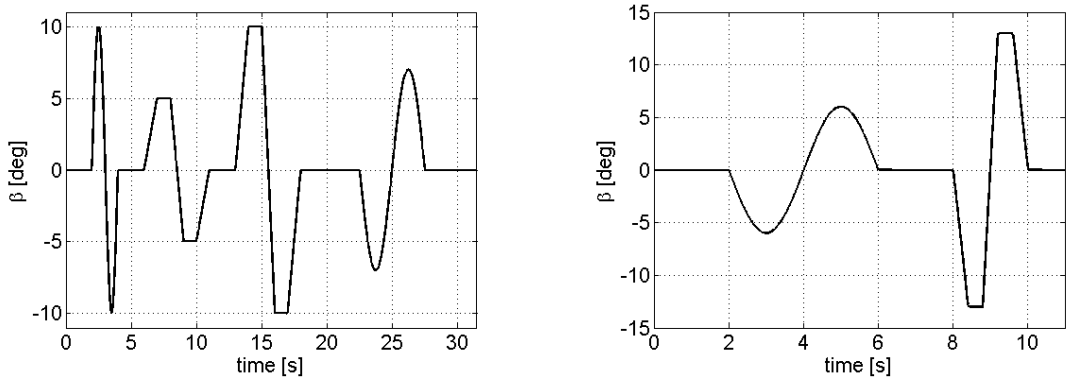
Linearity is the other aspect that should be considerd when using a linear formulation, such as (1.1). The couple of pressure ports at positions $P1$ and $P2$ have a more linear relationship, $\beta = f(\Delta C_p)$, than to $P3$ and $P4$ positions, in which the non-linear effects are enhanced. In particular, considering a linear approximation to describe function f of (3.42) at position $P3$ could lead to large errors when β is measured; as shown in Fig. 3.16(b), large errors ($e_\beta > 1.5\ deg$) result, even for limited lateral wind ($\beta \approx 7\ deg$). Moreover the error diverges for higher angle of sideslip higher than 12 deg.

However, these kinds of issues about positioning go beyond the real goal of this work, and a couple of static flush ports were considered in $P3$ to deal with a highly non-linear problem.

In order to model the problem of (3.42) using neural networks, training and test data must be defined. The training maneuver consist of sinusoidal and hold maneuvers, as a result of successive steady states, since unsteady CFD maneuvers are not performed. The maximum value of β reached during the training stage was 10 deg, see Fig. 3.17(a).

The test maneuver is made up of a sinusoidal sweep within the training boundaries, but at a different frequency than those used for training and of one β hold maneuver

whit a maximum value of 13 deg. The test maneuver was built in this way to evaluate



(a) Training maneuver (b) Test maneuver

Figure 3.17: Profiles of β used for the training and test stages

the ability of the NNs to generalize within the training boundaries and extrapolate outside.

Training technique selection

The BP and LM training algorithm are here compared in terms of speed of convergence. As known, the BP algorithm needs more iterations to reach a local minima than a second order algorithm. However, although the LM needs more time to solve each iteration, it needs less time to reach convergence than the back propagation algorithm. For this reason, the Levenberg-Marquardt algorithm was be used throughout this work to train the neural networks. In order to define the neural network architecture, two strategies were considered: pruning and growing.

Pruning technique

When the pruning technique is used to optimize NN performance, an oversized network must initially be considered. In this case, a feed-forward single-layer perceptron neural network is designed with 10 hidden neurons, one linear output neuron and two inputs, as depicted in Fig. 3.18, in order to represent the MISO system of (3.42). The results of the test maneuvers are reported in Fig.3.19(c). It is clear that these results are unacceptable and a neural network architecture optimization is required.

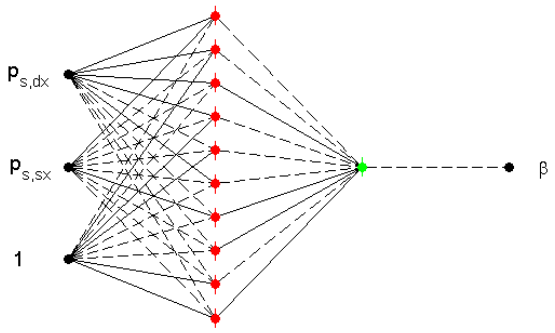


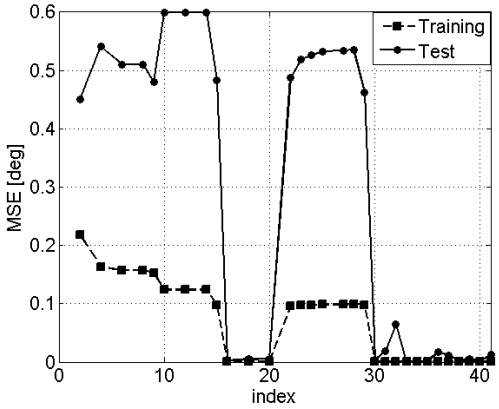
Figure 3.18: Neural net scheme. The solid lines represent positive synaptic weights, the dashed lines represent the negative synaptic weights.

The pruning technique, as described in [48, 50], is here used to reduce errors on test maneuvers; the results are presented in Fig. 3.19.

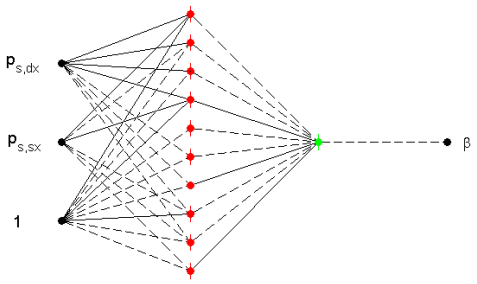
The pruning technique searches for those "weak" synaptic links and *prune* those synaptic weights, strictly speaking, setting them to zero, with the aim of building a more efficient neural network. After every cut, the neural network is re-trained for a few epochs, half the training ones at the most. As it is well known, the pruning process takes quite a long time: in this very light test case, it takes about 4 minutes. The mean squared error of training and test are reported in Fig. 3.19(a) for each iteration of pruning process. The neural net stored in index 34 shows the lowest MSE for the training and test patterns. The corresponding neural network is depicted in Fig. 3.19(b), where it can be noted that some synaptic weights are cut off, compared to that of Fig. 3.18. The improvement in NN performance is clear when the results of Fig. 3.19(d) are compared with those obtained before pruning: the maximum error is reduced from 2.04 *deg* to 0.23 *deg*.

Growing technique

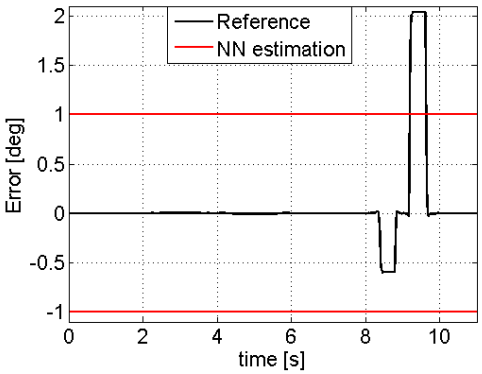
Unlike the pruning technique, when the growing approach is used, the initial neural network is undersized. Here, the starting NN has only one neuron in the hidden layer and it is then increased to ten. Several training processes are needed when this strategy is used, because the neural network must be retrained for each configuration



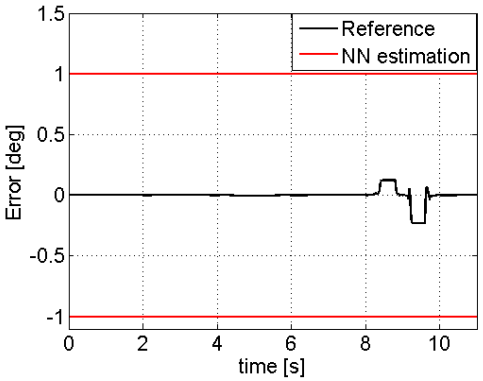
(a) Results of the pruning process



(b) Optimum configuration of the neural network



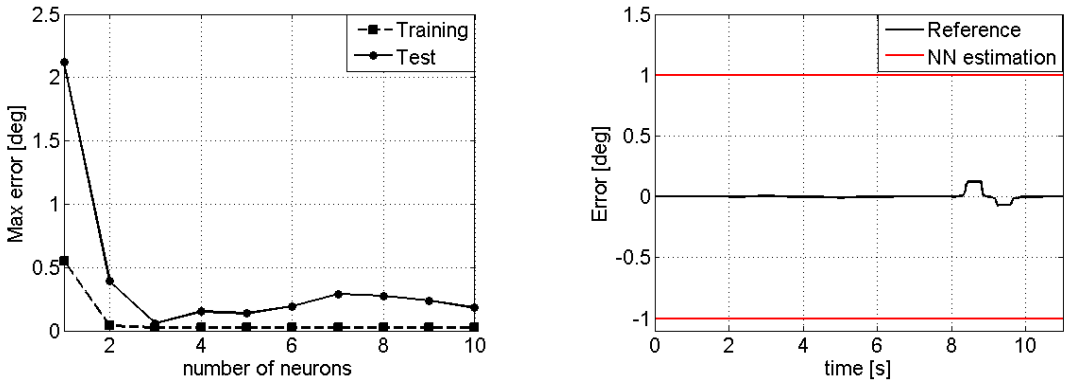
(c) Neural network performance before the pruning



(d) Neural network performance after the pruning

Figure 3.19: Results of the neural network pruning process

in order to avoid the local minima problem as described in 3.5.5. However, this kind of activity is performed with an automatic ad-hoc built Matlab routine which requires less time than the pruning approach for this particular case. From Fig. 3.20(a), it is clear that the best neural network should have three neurons inside the hidden layer, and that an increasing lack of generalization is experienced. The error profile of the β estimation is plotted in Fig. 3.20(b). The maximum error is only one-fourth, 0.06 *deg*, of that obtained after the pruning activity. The growing technique seemed faster and more reliable for this kind of activity, than the pruning one to optimize the neural network performance. The growing method was in fact used in all the applications presented in this work.



(a) Results of the growing process

(b) Neural network performance using the best NN configuration

Figure 3.20: Growing technique results

Chapter 4

Development of the NNs using the Matlab FDC toolbox

The validation of the virtual sensor based on neural network will be presented in this chapter. The aircraft model chosen for this purpose is the Beaver aircraft mathematical model which is available at *Mathworks*. This aircraft model was selected because its dynamics is comparable with the final application of virtual sensor: the actual Sky-Y UAV designed and developed by Alenia Aermacchi. The Beaver model was extensively used to define training and test maneuvers that could be representative of all flight conditions and, at the same time, short enough to be managed by a personal computer. In particular, in this chapter, attention is focused on the cruise phase and the approach and landing phases were neglected to avoid using flaps, because of they do not represent obstacles to training, as it will be shown in the next chapter.

The Matlab FDC toolbox was developed starting from the De Havilland DHC-2 "Beaver" aircraft [63] and it was designed to analyze non-linear aircraft dynamics and flight control systems. The aircraft is modeled as a rigid body with a constant mass value, flying in still air. The Beaver command system presents all the primary controls: the elevator command, δ_e, aileron command, δ_a, rudder command, δ_r, throttle command, δ_n, and the flap defelctions, δ_f. The thrust is adjusted by acting on δ_n which is proportional to the non-dimensional pressure increase in the propeller slipstream; for more details, reference can be made to [63].

4.1 Dynamic Analysis of the Aircraft

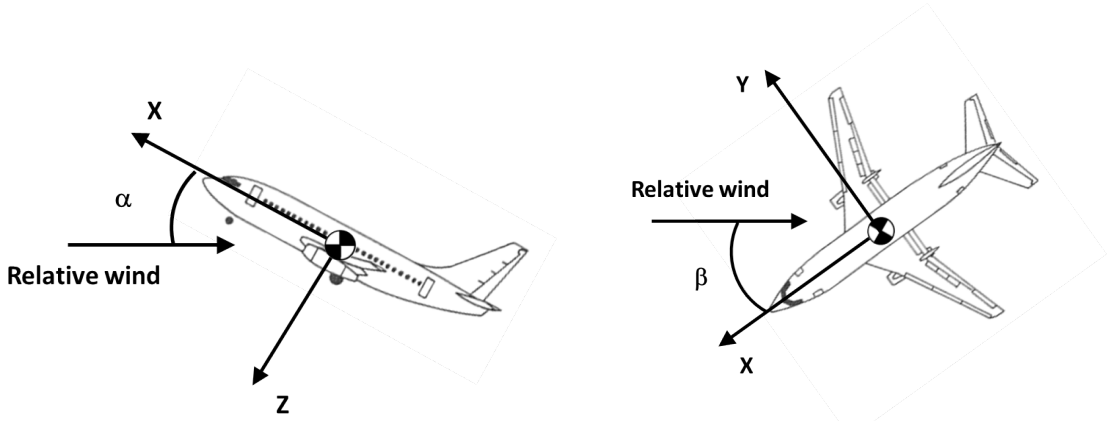
The aim of this section is to recall the differential equations of aircraft models, as they are commonly written [64] [65], in order to highlight the variables on which the aerodynamic angles depend.
Aircraft equations of motion are driven by aerodynamic, propulsive and wind forces and the moments acting at the center of gravity of the complete 6-DOF rigid aircraft [64]. If the pilot commands, which are obviously known, are fixed all the forces and moments depend on the aircraft orientation with respect to airflow, or conversely, on three quantities, defined as follows

- true air speed of the aircraft,V_∞ or TAS, with respect to the surrounding airflow;
- angle of attack, α, defined as the rotation, about the body y-axis, of the body-fixed x-axis needed to be aligned with the stability x-axis (see Fig. 4.1);
- angle of sideslip, β, defined as the rotation, about the body z-axis after the previous rotation, of the body-fixed x-axis needed to be aligned with the wind x-axis (see Fig. 4.1).

The angles of attack and sideslip are commonly indicated as *aerodynamic angles*.
The state vector is build using the velocity vector, Euler angles, angular rate vector and the position vector, as

$$X^T = [u,\ v,\ w,\ \phi,\ \theta,\ \psi,\ p,\ q,\ r,\ p_N,\ p_E,\ H]\,, \tag{4.1}$$

where H is the altitude in the NED reference frame. The aircraft height is usually referred to barometric height, calculated using the baro-altimeter. Today, the *GPS* height is used for navigation purposes, but, for safety reason, it is not used to separate the several flight levels in airways, because the barometric altitude error is much smaller than the geometric one.
The relationships between the body velocity vector of the state vector and the



(a) angle of attack (b) angle of sideslip

Figure 4.1: Definition of aircraft axes and angles

aerodynamic angles, can be expressed as follows

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = V \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix} \tag{4.2}$$

or, conversely

$$V = \sqrt{u^2 + v^2 + w^2} \tag{4.3a}$$

$$\alpha = \arctan\frac{w}{u} \tag{4.3b}$$

$$\beta = \arctan\frac{v}{\sqrt{u^2 + w^2}} \tag{4.3c}$$

Therefore, the state vector (4.1) can be rewritten as

$$X^T = [V,\ \beta,\ \alpha,\ \phi,\ \theta,\ \psi,\ p,\ q,\ r,\ p_N,\ p_E,\ h]\,. \tag{4.4}$$

The aerodynamic forces and moments depend on some state derivatives, and hence, the model is not linear and, referring to Fig. 3.5, can be described as

$$\begin{cases} \dot{X} &= f(X, U, v, w) \\ Y &= g(X, U, v, w) \end{cases} \tag{4.5}$$

where U is the command vector, which is defined as $U^T = [\delta_e,\ \delta_r,\ \delta_a,\ \delta_f,\ \delta_n]$.
Eq.s (4.3b), (4.3c) are differentiated in order to find the state derivative equations for aerodynamic angles, and the resulting equations constitute two of the complete set of ordinary differential equations which need to be solved in order to find the

analytical solution of the aircraft model for each time step.

Firstly, the angle of attack expression is developed, thus by differentiating the second equation of 4.3, it follows

$$\dot{\alpha} = \frac{u\dot{w} - w\dot{u}}{u^2 + w^2} \tag{4.6}$$

Inserting expressions of u and w 4.2 and rearranging the terms of (4.6) yields

$$\dot{\alpha} = \frac{\dot{w}\cos\alpha - \dot{u}\sin\alpha}{V\cos\beta} \tag{4.7}$$

According to [63], the relationships between the axial acceleration, gravity acceleration and angular velocities in the body reference frame can be written as

$$\begin{aligned} \dot{u} = &\quad rv - qw - g\sin\theta + \frac{F_x}{m} \\ \dot{v} = &\quad -ru + pw + g\sin\phi\cos\theta + \frac{F_y}{m} \\ \dot{w} = &\quad qu - pv + g\cos\phi\cos\theta + \frac{F_z}{m} \end{aligned} \tag{4.8}$$

where the contribution of the external force ($F_i = ma_i$) can also be indicated in terms of number of g acceleration of $n_i = a_i/g$, with $(i = x, y, z)$. Accelerometers are used to measure the absolute (inertial) accelerations, as the sum of the aerodynamic, propulsion and gravitational components. However, only in this section, the gravitational terms are removed from the body inertial accelerations, n_i, in order to better highlight how this contribution depends on the aircraft attitude, which is indicated as n_i'.

If Eq. 4.8 is considered, Eq. (4.7) can be rearranged as follows

$$\dot{\alpha} = \frac{(qu - pv + g_0\cos\phi\cos\theta + n_z')\cos\alpha}{V\cos\beta} - \frac{(rv - qw - g_0\sin\theta + n_x')\cos\alpha}{V\cos\beta} \tag{4.9}$$

Then, substituting (4.2), we can obtain the final expression of the ODE for the angle of attack, as

$$\begin{aligned} \dot{\alpha} = \frac{1}{V\cos\beta}\{ &[V\,(q\cos\alpha\cos\beta - p\sin\beta) + n_z' + g_0\cos\phi\cos\theta]\cos\alpha \\ &- [V\,(r\sin\beta - q\sin\alpha\cos\beta) + n_x' - g_0\sin\theta]\sin\alpha\} \end{aligned} \tag{4.10}$$

The same mathematical treatment can be followed to obtain the ODE for the angle of sideslip. The expression of the angle of sideslip expression is developed from (4.3). Thus by differentiating (4.3c), we obtain

$$\dot{\beta} = \frac{\dot{u}(u^2 + v^2) - v(u\dot{u} + w\dot{w})}{V^2\sqrt{u^2 + w^2}} \tag{4.11}$$

Substituting the u and w expressions (4.2), and rearranging the terms of (4.11 yields

$$\dot{\beta} = \frac{1}{V}(-\dot{u}\cos\alpha\sin\beta + \dot{v}\cos\beta - \dot{w}\sin\alpha\sin\beta). \tag{4.12}$$

If we substitute $\dot{u}$, $\dot{v}$ and $\dot{w}$ with the (4.8), the 4.12 can be rewritten as

$$\begin{aligned}\dot{\beta} = \frac{1}{V}[\,&(-n'_x + g_0\sin\theta + qw - rv)\cos\alpha\sin\beta + \\ &\left(n'_y + g_0\cos\theta\sin\phi + pw - ru\right)\cos\beta + \\ &(-n'_z - g_0\cos\theta\cos\phi + pv - qu)\sin\alpha\sin\beta].\end{aligned} \tag{4.13}$$

If the (4.2) is substituted in (4.13), some terms can be cancelled and the (4.13) can be rearranged as

$$\begin{aligned}\dot{\beta} = \frac{1}{V}[\,&(-n'_x + g_0\sin\theta)\cos\alpha\sin\beta + \left(n'_y + g_0\cos\theta\sin\phi\right)\cos\beta + \\ &(-n'_z - g_0\cos\theta\cos\phi)\sin\alpha\sin\beta] + p\sin\alpha - r\cos\alpha.\end{aligned} \tag{4.14}$$

Until this point, the treatment has had a general validity. In order to introduce an expression for F_i as a function of the pilot commands, or, more in general, as a function of the control surface deflections, it is necessary to characterize the aerodynamics of the particular aircraft. As far as the Beaver aircraft model [63] is concerned, the relationships between the inertial acceleration n_z, aerodynamics, propulsion and controls in (4.15), can be written as

$$\begin{aligned}n'_x = \frac{F_x}{m} = \frac{1}{2}\rho V^2(&C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_{\alpha^3}}\alpha^3 + C_{X_q}\frac{qc}{V} + \\ &C_{X_{\delta_r}}\delta_r + C_{X_{\delta_f}}\delta_f + C_{X_{\delta_e\alpha}}\delta_e\alpha + C_{X_{\delta_n}}\delta_n + C_{X_{\alpha\delta_n^2}}\alpha\delta_n^2),\end{aligned} \tag{4.15a}$$

$$\begin{aligned}n'_y = \frac{F_y}{m} = \frac{1}{2}\rho V^2(&C_{Y_0} + C_{Y_\beta}\beta + C_{Y_{\alpha^3}}\alpha^3 + C_{Y_r}\frac{rb}{2V} + \\ &C_{Y_p}\frac{pb}{2V} + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r + +C_{Y_{\delta_r\alpha}}\delta_r\alpha + C_{Y_{\dot{\beta}}}\frac{\dot{\beta}b}{cV}),\end{aligned} \tag{4.15b}$$

$$\begin{aligned}n'_z = \frac{F_z}{m} = \frac{1}{2}\rho V^2(&C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_{\alpha^3}}\alpha^3 + C_{Z_q}\frac{qc}{V} + C_{Z_{\delta_e}}\delta_e + \\ &C_{Z_{\delta_e\beta^2}}\delta_e\beta^2 + C_{Z_{\delta_f}}\delta_f + +C_{Z_{\delta_e\beta^2}}\delta_e\beta^2 + C_{Z_{\delta_n}}\delta_n).\end{aligned} \tag{4.15c}$$

However, for any kind of aircraft, it is always possible to find a relationship between the inertial accelerations, air data and pilot commands, which are obviously obviously grouped in specific ways for each particular aircraft model.

Eq.s (4.10), (4.14) are essential to understand the independent variables on which α and β depend. Combining Eq.s (4.10), (4.14) and (4.15), it is possible to write the following functional relationship

$$\alpha = F_\alpha \left(q_c, n_x, n_y, n_z, \beta, \theta, \phi, p, q, r, \delta_e, \delta_a, \delta_r, \delta_n, \delta_f\right) \tag{4.16a}$$

$$\beta = F_\beta \left(q_c, n_x, n_y, n_z, \alpha, \theta, \phi, p, q, r, \delta_e, \delta_a, \delta_r, \delta_n, \delta_f\right) \tag{4.16b}$$

where the velocity V (or TAS) has been substituted by the dynamic pressure because it is the source from which the velocity is derived, as described in 1.2, and the inertial accelerations are those calculated by the accelerometers.
As mentioned at the beginning of this section, the commands are not included in the state vector, but are external inputs. In this work, Eq.s (4.16) were approximated as

$$\alpha = f_\alpha \left(q_c, n_x, n_y, n_z, \beta, \theta, \phi, p, q, r, \delta_e, \delta_a, \delta_r, \delta_n, \delta_f\right) \tag{4.17a}$$

$$\beta = f_\beta \left(q_c, n_x, n_y, n_z, \alpha, \theta, \phi, p, q, r, \delta_e, \delta_a, \delta_r, \delta_n, \delta_f\right) \tag{4.17b}$$

in order to have the simplest virtual sensor that is able to satisfy the prescribed requirements. At this stage of the work, because of the non-linear aircraft dynamics is considered, we decided to keep all the variables. However, the relationship between each input and output will be discussed later in chapter 6.

4.2 Strategy for Creating Training and Test Maneuvers

The basic training maneuver designed for the Beaver simulator set presented in this section is the result of several attempts to build an effective training pattern that is able to give as much information as possible about the whole aircraft flight envelope. As previously mentioned in section 3.5, the most representative training data set should be used to train the neural network, but for certain applications, where a large amount of data must be managed, a strategy for data reduction is required if the hardware resources of a common workstation were designed to be sufficient.
The Simulink simulation of the Beaver aircraft requires a very small integration step (about 20 ms), therefore, the resulting flight data vectors are sampled at 50 Hz.

Since only off-line training is considered in this work, a training maneuver of 5 minutes, which contains 6000 points for each vector, and the complete input-output pattern, as described in the previous section, which has at least 9 inputs and one output, means a training set is obtained that contains about 60000 points. In order to cover the entire flight envelope, several maneuvers are repeated for several velocity from stall velocity without using flaps ($\simeq 85\ kts$) to maximum velocity ($\simeq 130\ kts$) in level flight. The strategy of repeating a basic training maneuver at several velocities was used for this work. During this work, it was noted that the neural network was able to extrapolate adequately between velocities if separated by $10 - 20\ kts$. This was the first step taken to reduce the training set. Moreover, the input vectors were not used at $f = 50\ Hz$ but they were sampled at lower frequency in order to further decrease the number of data in the training vectors without removing information about the aircraft dynamics.

Since the ADSs of unmanned aircraft are calibrated during the first flight tests, no high-dynamic maneuvers are performed for safety reasons. The same conditions were considered to define a training strategy: several dynamic maneuvers, clearly distinguished from each other to avoid couple longitudinal, lateral and directional dynamics as much as possible, were performed at almost constant velocity. Some time was necessary after each maneuver waited to re-establish the initial stabilized flight conditions. In this way, a basic training maneuver set is created. As also described in section 1.2.1, the basic maneuver set should be made up of maneuvers that are able to excite aerodynamic angles for the entire working range. Unfortunately, under the realistic hypothesis of a first flight test, and therefore calm maneuvers, the extension is left, if possible, to neural network extrapolation capabilities, as it will be shown during the test stage. A flight card was prepared before any training campaign in order to give the autopilot a well prescribed flight pattern to follow using an ad hoc written Matlab routine. A general guideline for training maneuvers is reported in the Fig. 4.2, where a basic maneuver set is performed for each velocity. The *hold* maneuvers are performed with the aim of collecting information on stabilized flight conditions, while the *sweep* maneuver is designed to excite the short period mode of the airplane. Maneuvers are executed using the MIMO autopilot available

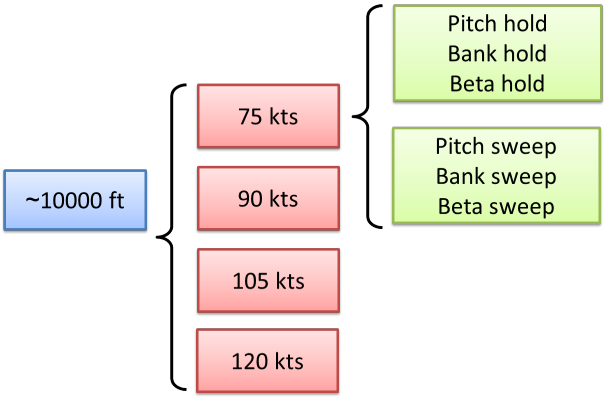


Figure 4.2: Guideline scheme for the training maneuver set

Maneuver	Limits	Duration
Pitch hold	$\pm 5\ deg\ (\theta)$	$10 - 20\ s$
Bank hold	$\pm 20\ deg\ (\phi)$	$10 - 20\ s$
Beta hold	$\pm 5\ deg\ (\beta)$	$10 - 20\ s$
Pitch sweep	$\pm 5\ deg\ (\theta)$	$< 10\ s$
Bank sweep	$\pm 5\ deg\ (\phi)$	$< 10\ s$
Beta sweep	$\pm 20\ deg\ (\beta)$	$< 10\ s$

Table 4.1: Details of the basic training maneuvers

in the FDC toolbox, suitably modified to perform maneuvers within the prescribed limits, which are reported in Tab. 4.1. The test maneuver set is composed of several high-dynamic maneuvers. The three aircraft dynamics are often mixed with speeds ranging from the stall speed, without flap, to the maximum speed obtainable during dive ($\simeq 130\ kts$), in order to evaluate the neural network performance in extreme conditions and outside the training boundaries.

During test maneuvers (see Fig. 4.5), no limits are prescribed to single maneuvers in order to evaluate the real neural network performance. In particular, the pitching motion is combined with yawing and rolling maneuvers in order to evaluate the performance of the neural network in estimating superimposed motion that is not performed during the training stage. Then, moving the throttle to maximum power, pushover and pullup maneuvers are performed to evaluate the results of the virtual sensors when very low and very high dynamic angles of attack are achieved during unsteady flight conditions. Obviously, the conditions described here are quite extreme and sometimes outside the real aircraft flight envelope, but they were very useful in order to evaluate which is the best neural net configuration able to learn and

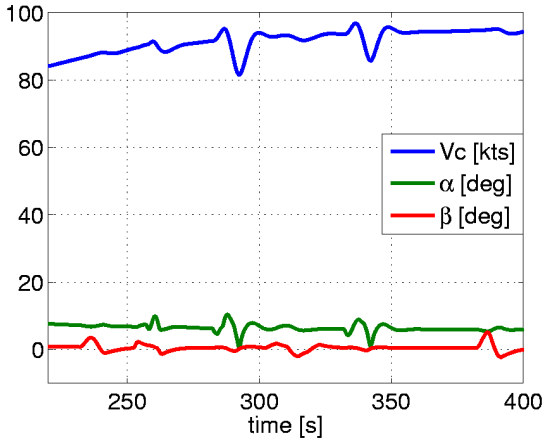


Figure 4.3: A basic training set maneuver performed at 90 *kts*

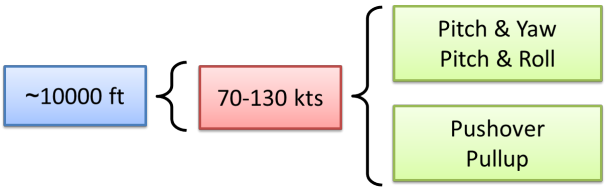


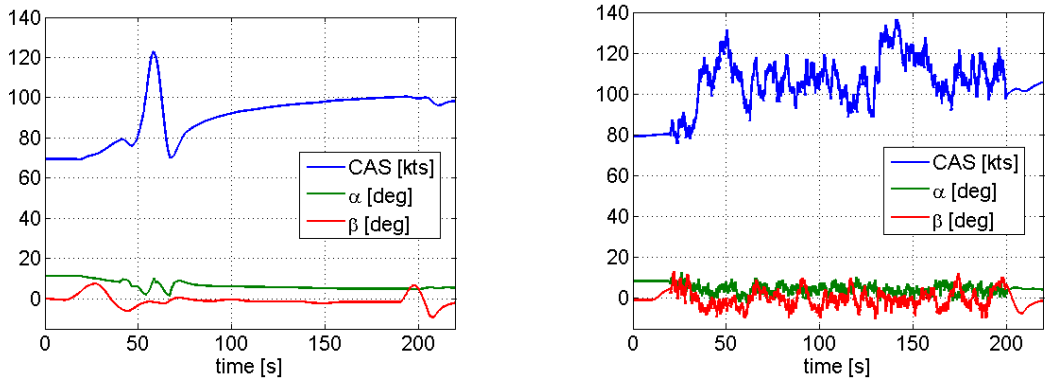
Figure 4.4: Guideline scheme for test maneuver set

generalize Beaver dynamics. During test maneuvers several levels of turbulence [66, 66] and wind gusts were simulated, unlike the training strategy, to increase the gap between the training and test pattern in order to conduct a suitable study of NN performance, capability to generalize within the training boundaries and to extrapolate outside the training boundaries.

4.3 Definition of the Neural network Architecture

In this section, the neural network architecture of the virtual sensors will be presented as the result of several attempts to obtain the best performance at the lowest computational cost. In other words, this stage may be considered the most important for virtual sensor design. In fact, since the final application of the neural network is for Sky-Y UAV which has a similar flight envelope and dynamics to Beaver aircraft, the Beaver simulator was used to set up the neural networks and to study its sensitivity to each of the parameters.

At the beginning, a feed-forward single layer perceptron neural network was used



(a) Without turbulence

(b) With severe turbulence(probability = 10^{-6} [66])

Figure 4.5: Test maneuvers

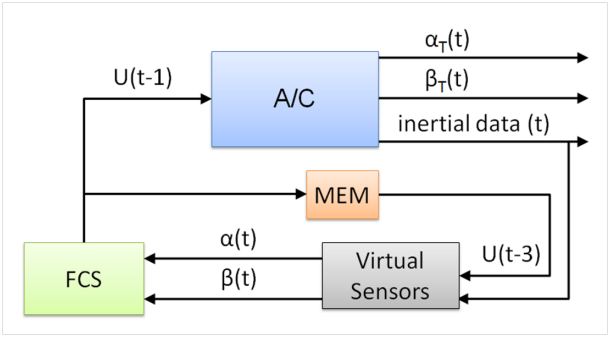


Figure 4.6: Integration of virtual sensors (NN) with the aircraft control system (indicated as $AFCS$)

to compare the results during the designing stage of NN architectures. After a brief analysis of the control system, described briefly in Fig. 4.6, the block scheme was introduced in order to highlight the relationships between the inputs and outputs of the Beaver aircraft and to avoid any closed loop that could introduce instabilities into the overall control loop design. As can be seen in Fig. 4.6, the inertial data (attitudes and rates) and the velocity (acquired from measurements of a Pitot-static device) can be used in the present time frame, t, while the surface deflections must be considered at least one time step back, to avoid a loop that closes on itself. According to these considerations, the available input data for the virtual sensors are:

- dynamic pressure, $q_c(t)$
- longitudinal body acceleration, $n_x(t)$
- lateral body acceleration, $n_y(t)$
- vertical body acceleration, $n_z(t)$
- pitch angle, $\theta(t)$
- roll angle, $\phi(t)$
- body roll rate, $p(t)$
- body pitch rate, $q(t)$
- body yaw rate, $r(t)$
- aileron deflections, $\delta_a(t-3)$
- elevator deflections, $\delta_e(t-3)$
- rudder deflections, $\delta_r(t-3)$
- flap deflections, $\delta_f(t)$

All the listed inputs are considered without the subscript that indicates free stream, because they are considered measured by dedicated probes or sensors. As far as the dynamic pressure measurement is concerned, the ISA atmosphere was considered for flight simulations in this work. Therefore the static temperature was known at any

altitude and it was easy to calculate the true air speed from the dynamic pressure. However, since the aerodynamic coefficients are functions of the (measured) true dynamic pressure, q_c, only the use of true dynamic pressure is considered for this work, even if it is always possible to convert q_c into TAS when a temperature measure is available aboard (see section 2.2), the use of dynamic pressure resulted to be easier than TAS and did not need any additional sensor for temperature measurements. Moreover, the flaps were not considered for the *Beaver* application. The starting neural net architecture (see Fig.) has

- all the available signals in the input layer,
- a variable number of non-linear neurons in the single hidden layer,
- 1 output layer.

As described in section 3.5.6, the results that will be presented in this work are the results of several (usually more than 7) re-initializations that were conducted in order to avoid any local minima. A first sensitivity analysis is reported in table 4.2, where the performance of the neural nets used for angle of attack estimation (NNA) are shown for changes in the input vector. As will be shown later on, some

Input	Performance
all	$2.5e^{-6}$
without q_c	$6.1e^{-4}$
without n_z	$2.3e^{-5}$
without θ	$1.8e^{-5}$
without δ_e	$3.4e^{-5}$

Table 4.2: Performance analysis of NNA without some inputs

important preliminary results were obtained. The virtual sensor must be fed with velocity and inertial data, while the command can also be neglected for angle of attack estimation. The same preliminary sensitivity analysis was performed for the neural network, which was used for the estimation of the angle of sideslip (NNB), and it is reported in table 4.3. Again, velocity and inertial data are the most important parameters, but the commands cannot be neglected for the angle of sideslip.
The growing technique described in section was used in order to define the best neural network architecture. As known from previous applications, the number of

Input	Performance
all	$1.2e^{-5}$
without q_c	$7.2e^{-4}$
without n_y	$3.7e^{-4}$
without ϕ	$4.6e^{-4}$
without δ_r	$5.5e^{-4}$

Table 4.3: Performance analysis of NNB without some inputs

neurons must be equal or larger than the number of input signals number, but not too large so that neural networks can generalize.

The performance of the virtual sensor for the angle of attack estimation is reported in table 4.4. It was noted that at over 15 neurons with a single layer neural network

Number of neurons	Number of hidden layers	Training mse $[deg]$	Max error on training $[deg]$	Max error on test $[deg]$
5	1	$4.9e-5$	0.19	1.39
5	2	$4.6e-6$	0.15	2.32
10	1	$5.7e-6$	0.083	1.12
10	2	$6.6e-7$	0.067	1.53
15	1	$3.0e-7$	0.013	0.626
15	2	$1.6e-7$	0.010	0.67
20	1	$3.5e-7$	0.013	0.853
20	2	$1.0e-7$	0.0090	1.31
25	1	$7.6e-8$	0.0096	0.880
25	2	$2.5e-8$	0.0080	1.30

Table 4.4: Error performance of NNA, varying the number of neurons and hidden layers

the maximum errors obtained during test maneuvers were not further improved (see ; while, as far as the training stage is concerned, the better the performance, the higher the number of neuron used. This particular behaviour, i.e. the NN performance of the training pattern is improving while the NN performance on the test pattern is decaying, is typical when NNs memorize the training data set but do not "learn" the system dynamics, and hence, in this case, the NNs are able to generalize.

The same conclusion can be reached for the angle of sideslip.

In occlusion, for this kind of application, the use of a single layer neural network with fifteen non-linear neurons showed to be the best compromise between prediction

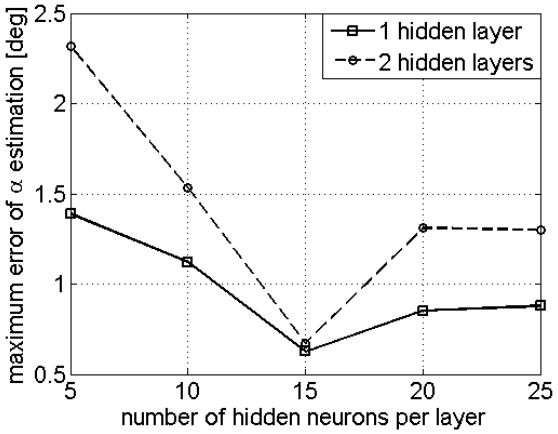


Figure 4.7: Maximum error for the angle of attack estimation during the test maneuvers

accuracy and hardware complexity.

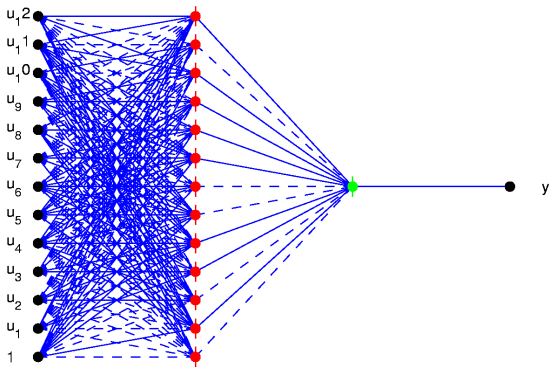


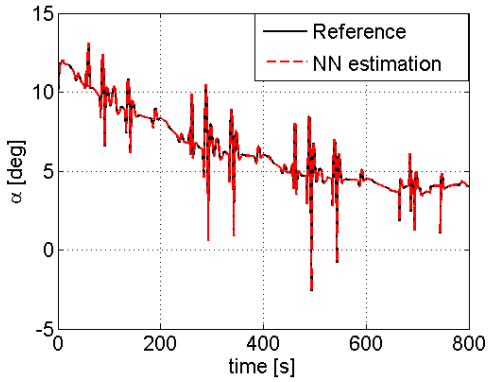
Figure 4.8: Starting layout of neural network architecture

4.4 Virtual Sensor Performance

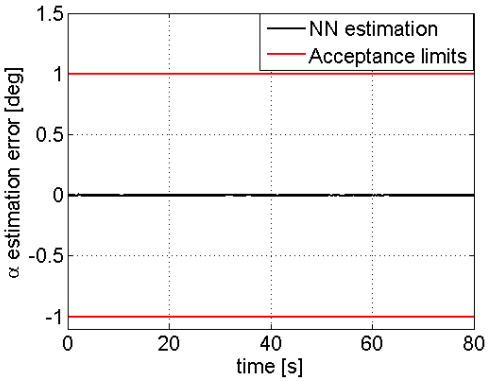
The results of the application of the virtual sensor to the Beaver aircraft model are shown in this section, in order to evaluate the actual neural network performance for both the angle of attack and sideslip. As stated in the previous section, the same neural network architecture, depicted in Fig. 4.8, was used for both aerodynamic angles.

4.4.1 Angle of Attack

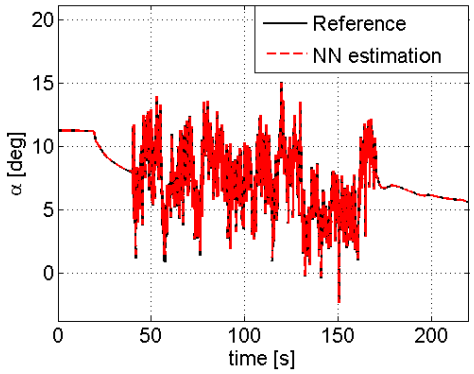
The reference angle of attack profile is reported in Fig. 4.9 with a solid line, while the virtual sensor prediction is plotted using a dashed line. This plot format is used throughout this work. The two curves are very close and the maximum error is close to zero degrees, as can be noted in Fig. 4.9(b). The free stream, or reference angle of attack registered during the test maneuver is presented in Fig. 4.9(c), where the virtual sensor estimation is also plotted. The two curves remain very close during all the simulations, except for some frames, where the estimation error is not negligible, even though it is always within the acceptance limits, as reported in Fig. 4.9(d). By analyzing the error profile, it can be seen that the maximum estimation error ($\simeq 0.8\ deg$) occurs when the throttle is pushed quickly to the maximum position and the rudder and elevator were actuated to yaw ($\beta = \pm 6\ deg$) and pitch ($\alpha = \pm 9\ deg$) simultaneously. The neural network designed for the α estimation for the Beaver simulator was accurate enough and it was therefore used as the starting point for



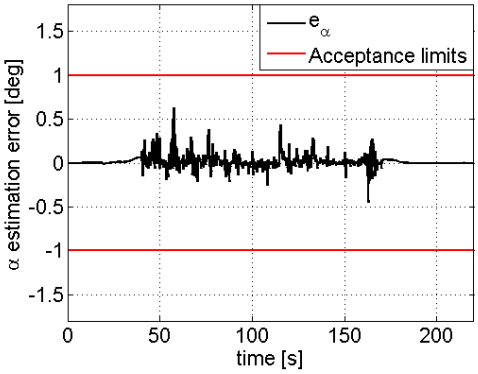
(a) Reference and estimated angle of attack during training maneuver



(b) Error of angle of attack estimation during training maneuver



(c) Reference and estimated angle of attack during test maneuver



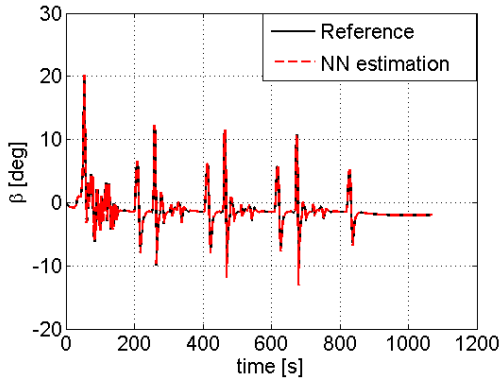
(d) Error of angle of attack estimation during test maneuver

Figure 4.9: Virtual sensor performance in Beaver angle of attack estimation for training and test maneuvers

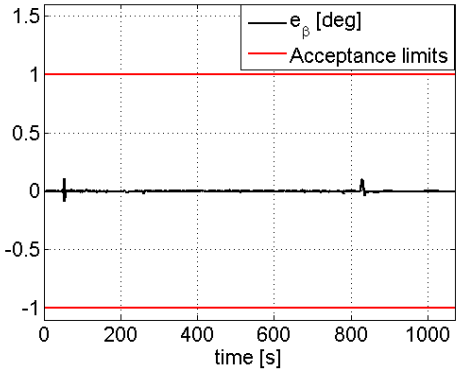
the Alenia Aermacchi Sky-Y applications.

4.4.2 Angle of Sideslip

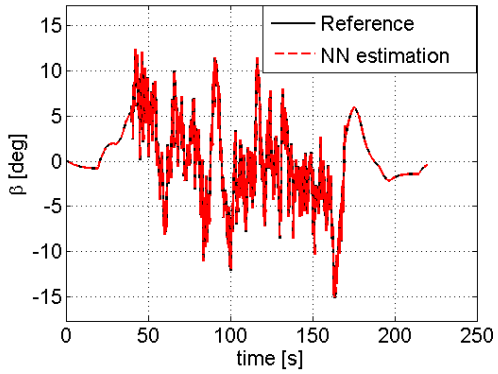
The reference angle of sideslip and neural network prediction profiles for the training and test maneuvers are reported in Fig. 4.10. The reference and estimated profiles overlap almost perfectly in Fig. 4.10(a) and the maximum error is almost null, as can be seen in Fig. 4.10(b). The reference angle of sideslip collected during the test maneuver is presented in Fig. 4.10(c), where the virtual sensor estimation is also plotted. Again, the two curves are very close during the entire simulations, and the errors of β prediction are negligible even though severe turbulence was simulated, as reported in Fig. 4.10(d). Overall, the NN designed for the β estimation of the Beaver



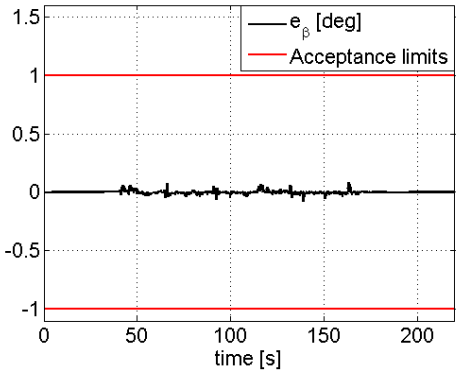
(a) Reference and estimated angle of sideslip during training maneuver



(b) Error of angle of sideslip estimation during training maneuver



(c) Reference and estimated angle of sideslip during test maneuver



(d) Error of angle of sideslip estimation during test maneuver

Figure 4.10: Virtual sensor performance in Beaver angle of sideslip estimation for training maneuver

application is very accurate: the maximum estimation error is less than one-tenth of all the maneuvers considered here, during which high dynamics and turbulence were simulated. Therefore, the NN designed here for the angle of sideslip, was used as starting the architecture for the Sky-Y simulator.

4.4.3 Simulating Real Flight Instrument Noise

In the previous section the real external world has been represented by turbulence and wind. In this section, the reality of the test campaign is augmented by also considering the electronic noise on each of input signals from its corresponding sensor.

General speaking, the actual inertial signals, if compared to simulated signal, are

Variable	Noise level (peak to peak)
q_c	$\pm 0.03\ mBar$
n_x	$\pm 0.0085\ g$
n_y	$\pm 0.0085\ g$
n_z	$\pm 0.0085\ g$
θ	$\pm 0.01\ deg$
ϕ	$\pm 0.01\ deg$
p	$\pm 0.1\ deg/s$
q	$\pm 0.1\ deg/s$
r	$\pm 0.1\ deg/s$

Table 4.5: Noise level used to corrupt the mathematical model data

always affected by noise that comes from several sources, such as engine induced vibrations and on board sensor electronic noise. The electronic noise is here represented using the data specifications of commercial MEMS Attitude Heading Reference Signals (AHRS) [67, 62]. The electronic noise of dynamic pressure, q_c, was modeled according to some available data sheets [68]. Usually, the electronic noise is given as the density over the square root of frequency ($1/\sqrt{Hz}$) in data sheets, as depicted in Fig. 4.11 for inertial acceleration measurements. The solid line is the noise density

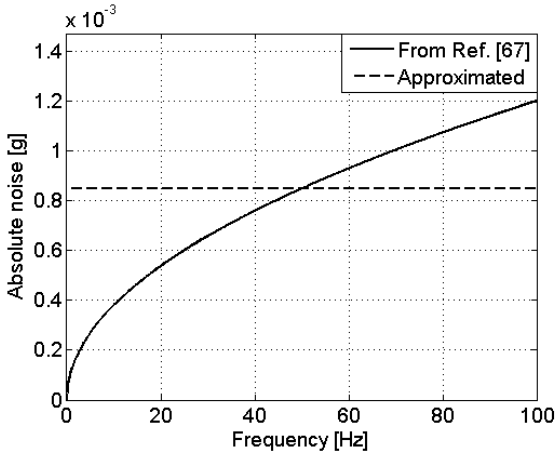


Figure 4.11: Example of noise density for inertial accelerations. The profile from data sheet [67] and the approximation used in this work are plotted

profile measured by the manufacturer, while the dashed line is the white noise level chosen for the present test, where the maximum value was selected at 50 Hz, in agreement with the refresh rate here considered, to establish the peak-to-peak level reported in Tab. 4.5. A white noise, with a zero mean value, was added to each

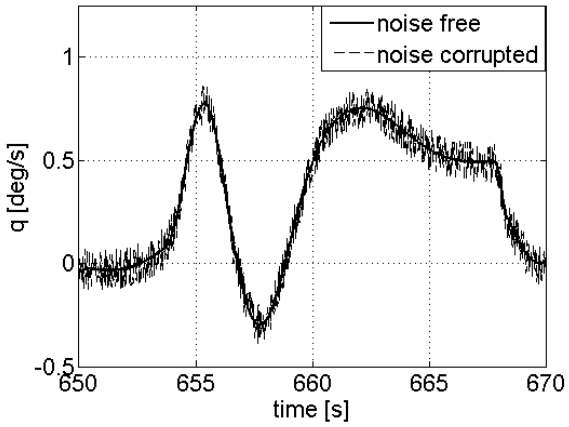
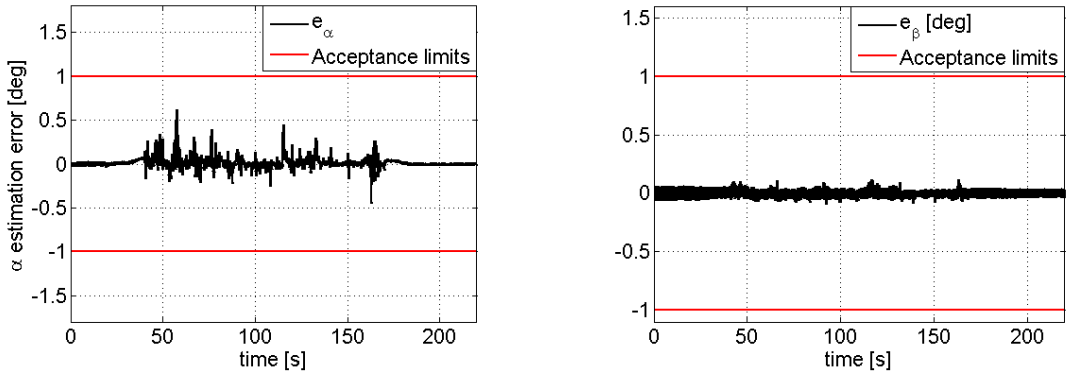


Figure 4.12: Example of pitch rate with and without noise

signal and the peak-to-peak values in Tab. 4.5 were selected according to available literature [67, 68, 69, 70]. Electronic noise on the surface deflection signals was neglected because it was of the order of magnitude of 0.001 deg, which was less than the resolution of considered actuators. Fig. 4.12 shows an example related to the pitch rate, q.

The same neural network trained at the end of previous section was used here for testing, using input signals corrupted with white noise in order to represent the worst case of real sensors.

The same NNs described in the previous section were tested using noisy input signals. The error profile of neural network estimation for the angle of attack during test maneuvers is plotted in Fig. 4.13(b). Fig. 4.13(a) shows the error in α prediction



(a) Angle of attack

(b) Angle of sideslip

Figure 4.13: Virtual sensor performance in the Beaver angles of attack and sideslip estimation for test maneuvers using noisy input signals

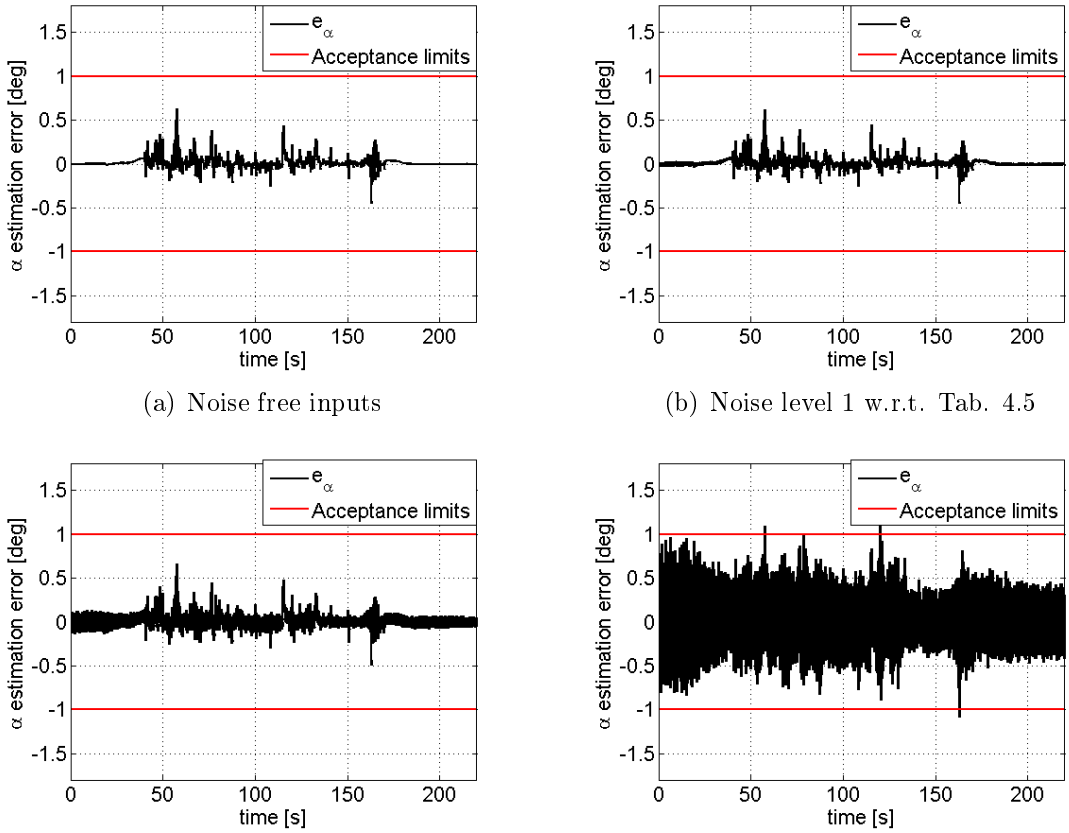
for a test maneuver set. The use of input signals corrupted with artificial noise led to error of α prediction of less than half degree.
As far as the β estimation is concerned, Fig. 4.13(b) shows that the maximum error is limited to 0.5 *deg* like that of the angle of attack.
In conclusion, the noisy inputs introduced additional errors (results in the previous section) less than one-tenth degree, if compared with results obtained using noise-free inputs. The electronic sensor noise can therefore be neglected for this particulate NN application because did not introduce significant disturbances.

4.4.4 Impact of Noisy Inputs on Neural Network Performance

The degradation of performance of NN-based virtual sensors, which is occurs when noisy data are used instead of simulated data, was analyzed in this section. As discussed in the previous section, the electronic noise is adequately managed by the NNs, since the aerodynamic angle predictions are always within tolerance limits, even in the worst conditions. If the noise is increased several times, the NN estimations degrades beyond the acceptance limits established at the beginning of this work.
The inertial measurements could be highly influenced by other stimuli which are not the sole aircraft acceleration or angular rates. The presence of external disturbances contributed to create corrupted signals which were considered here as another source of noise for the NN input signals. The additional disturbances mainly stem from structural vibrations of the aircraft.
As far as structural vibration is concerned, the vibration frequency spectra of real aircraft consists of a broadband background with superimposed narrow band spikes. The background spectrum results mainly from the engine, combined with many lower level periodic components, due to the rotating elements (engines, gearboxes, shafts, etc.) associated with the propeller, and some specifications were collected in ref. [71] for several aircraft categories. The vibration sensed by onboard inertial instruments, such as AHRS, depends on the kind of material used for the aircraft structures, the method adopted to isolate the structural vibrations, and so on. The discrepancies between the mathematical model and the real aircraft can be considered as another source of noise, especially when the sensor sampling is close to the first structural

mode frequencies, such as in slender bodies. Therefore, the noise characterization depends on the particular aircraft at hand and it should be treated case by case using, for example, ad hoc notch filters, or other non-conventional filters, e.g base on neuro-fuzzy (NF) techniques, which do not introduce time delays []. However, this kind of discussion is related to special cases and does not have a general validity and is therefore beyond the main focus of this document.

In order to estimate the noise level beyond which the NN estimations were not acceptable, the uncertainty propagation method [72, 73] was considered here; several levels of noise were therefore tested. The peak-to-peak noise levels in Tab. 4.5 were doubled, triplicated and so on until the acceptance limits were passed.
A comparison of the error profile of NN estimation for the angle of attack, α, is shown in Fig. 4.14 when several input vector noise levels were considered. A noise



(a) Noise free inputs

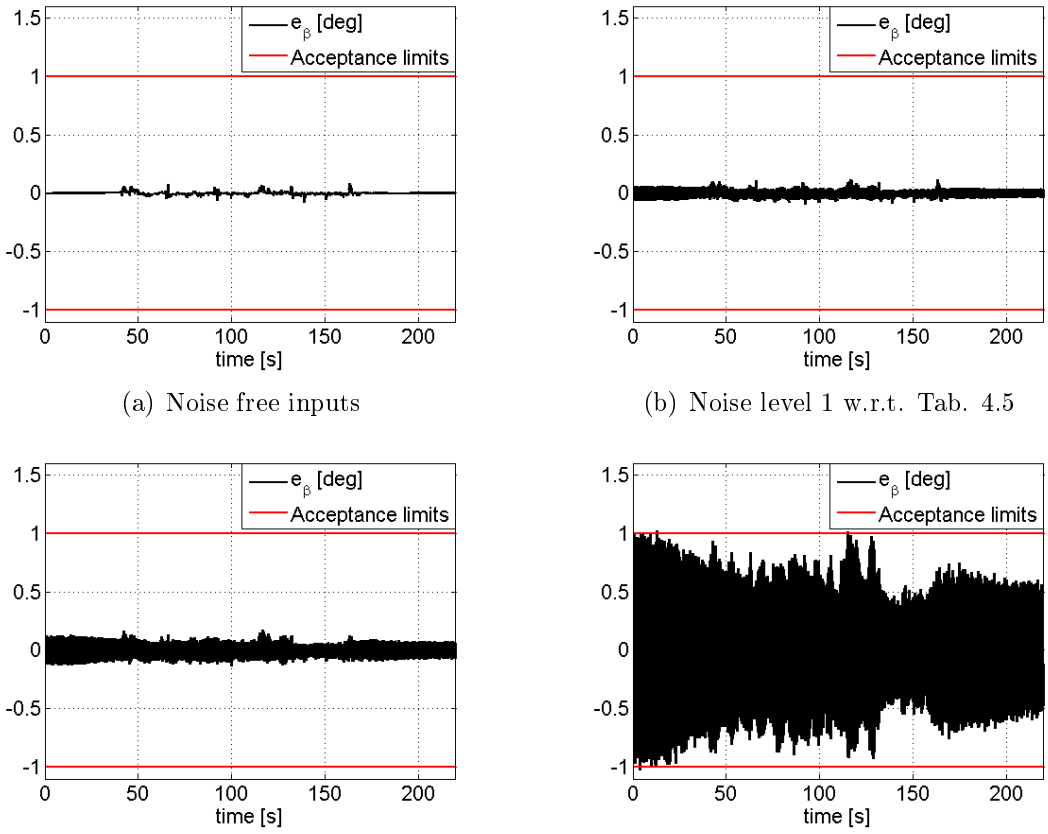
(b) Noise level 1 w.r.t. Tab. 4.5

(c) Noise level 5 w.r.t. Tab. 4.5

(d) Noise level 34 w.r.t. Tab. 4.5

Figure 4.14: Virtual sensor performance of the angle of attack estimation for the Beaver application during test maneuvers with several noise levels on inputs

level below 34 (Fig. 4.14(d)) did not cause errors greater than ±1 *deg*. A noise level of five times, with reference to those in Tab. 4.5, produced additional errors on NN estimations bounded within ±0.1 *deg*, as shown in Fig. 4.14(c), if compared with the NN α estimations obtained with noise-free input signals (Fig. 4.14(a)).
Error profiles of angle of sideslip estimation when the input vector is corrupted with several levels of noise are shown in Fig. 4.14. A noise level below 16 (Fig. 4.15(d))



(a) Noise free inputs

(b) Noise level 1 w.r.t. Tab. 4.5

(c) Noise level 2 w.r.t. Tab. 4.5

(d) Noise level 16 w.r.t. Tab. 4.5

Figure 4.15: Virtual sensor performance of the angle of sideslip estimation for the Beaver application during test maneuvers with several noise levels on inputs

did not cause errors greater than ±1 *deg*. A noise level of five times, with reference to those in Tab. 4.5, produced additional errors on NN estimations bounded within ±0.1 *deg*, as shown in Fig. 4.15(c), if compared with the NN β estimations obtained with noise-free input signals (Fig. 4.15(a)).
Overall, wherever the noise comes from, if the its spectra is below 16 times of those presented in Tab. 4.5, the virtual sensors were able to predict both α and β with an error less than ±1 *deg*; over 16 times and up to level 32, only the NNA was still able

to estimate α with acceptable accuracy, unlike NNB. By comparing results obtained for β with those obtained for α, is therefore clear that the neural network for the angle of sideslip estimation was more sensitive to noise on input signals than neural network designed for angle of attack estimation. This preliminary noise-sensitivity analysis is extended in the chapter 6 and disbudded with more details.

Chapter 5

Neural Network Test on the Alenia Aermacchi Sky-Y UAV Integration Rig

The Sky-Y aircraft is an unmanned airplane that was designed by Alenia Aermacchi for medium altitude and long endurance (*MALE*) for demonstration and research purposes.

Figure 5.1: The Alenia Aermacchi Sky-Y

Some of the specifications of the aircraft are:

- length: 9.7 m
- wing span: 9.9 m
- MTOW: 1200 Kg
- cruising speed: 140 kts

- ceiling altitude: $> 25000 ft$
- endurance: 14 h

Sky-Y is a dedicated UAV platform that cab be used to validate several key enabling technologies so that surveillance unmanned uerial uystem (UAS) can be used in either military or civil operational scenario. These thechnologies include: innovative carbon fiber composite construction, heavy *fuel/JP-8* engine (automotive diesel derivative), advanced datalinks, surveillance sensor (EO/IR, Hyperspectral, Synthetic Aperture Radar) and a mission management system that is able to relevant data treatment, elaboration, fusion and distribution by means of an interoperable Tactical Control Station. Sky-Y, thanks to its all-composite structure and a 160 HP diesel engine, is able to fly up to 12 hours. On 30 October 2007 it set a new European endurance record in the over 1000 kg category with a spotless eight-hour flight.
Simulators are usually used extensively in order to train UAV pilots. The Sky-Y simulator has been designed by Alenia Aermacchi with the aim of training pilots in UAV dynamics and of making them more familiar with ground control stations. For this work, the simulator was used to train and test the neural network developed for Sky-Y UAV for aerodynamic angle estimation. The NN tests, which were perfromed using the Sky-Y simulator, were then extended for validation on real hardware, with the aim of validating the NN software on the real FCC of the Sky-Y aircraft.
The main benefit of using the Sky-Y simulator, instead of real flight data, is the possibility of studying virtual sensors in extreme conditions that cannot be flown during Sky-Y flight tests for safety reasons, for example during severe turbulence conditions or at speeds higher than V_{NE} up to V_D. This is useful to investigate performance degradation beyond the training limits, as will be shown in this chapter. The knowledge of neural network functioning, beyond the training boundaries, is one of the most important aspects for real world use, since it allows engineers to prove that neural net performances are acceptable in all flight conditions and even during maneuvers that the aircraft will never fly. In particular, the understanding of the neural network operating behaviour, even when the training limits were exceeded, could be one of the key points to a successful certification, as will presented briefly at the end of this chapter. First, the training and test maneuvers performed

on the Sky-Y simulator will be presented in detail. The performance of virtual sensors designed for angles of attack and sideslip will be documented using both the Sky-Y simulator, simulating sensor noise, and the real flight control computer in the simulation loop. Finally, neural networks will be tested using real data collected during the Sky-Y flight test campaign.

5.1 Training and Test Maneuvers

The training and test strategies for the virtual sensors adopted for this activity were the same as those presented in the previous chapter: a basic training maneuver set was repeated at several velocities in order to cover the entire flight envelope (from minimum speed with flap to maximum speed). The test consisted of several high dynamic and full authority command maneuvers which exceeded aerodynamic angle limits flown during the training stage. For the training stage of the Sky-Y simulator activity, several dynamic maneuvers were performed at almost constant velocity, each maneuver being clearly distinguished from the others in order to avoid coupling longitudinal, lateral and directional dynamics as much as possible. Moreover, for this purpose, an interval of time was left between maneuvers in an attempt of to re-establish the initial level flight conditions. This is very important for two reasons. Firstly, to represent the same flight conditions encountered during the first flight tests; secondly, to train the neural networks with low-dynamic maneuvers in order to evaluate, during the validation or test stage, the neural network ability to estimate aerodynamic angles, even during high-dynamic maneuvers. This latter point is really a key factor in neural network design. In fact, since it is almost impossible to foresee any flight conditions of a real life aircraft envelope during the training stage and, even if it were possible, there would be hardware resource issues regarding the management of such an amount of data. It was therefore very important to investigate the extrapolation and generalization performance of neural networks.
A flight card was prepared for simulations before each training campaign,, in order to give the pilot a well prescribed flight pattern to follow. A general guideline for training maneuver is reported in the Fig. 5.2, where a basic maneuver set was

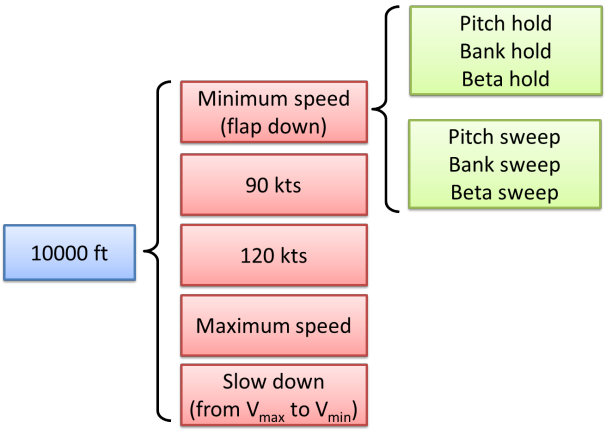


Figure 5.2: Guideline scheme for training maneuver set

performed by pilots aided by auto-pilot modes for each velocity.

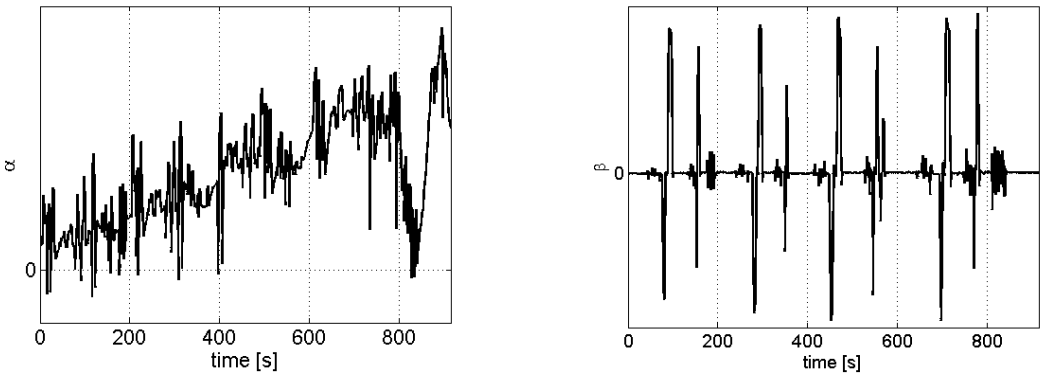
The basic maneuver set was made up of six maneuvers: pitch hold, bank hold, beta hold, pitch sweep, bank sweep and beta sweep. Indications were provided to the pilot for each one, as reported in table 5.1. The *hold* maneuvers are performed

Maneuver	Limits	Duration
Pitch hold	$\pm 5\ deg\ (\theta)$	$10 - 20\ s$
Bank hold	$\pm 20\ deg\ (\phi)$	$10 - 20\ s$
Beta hold	$\pm 5\ deg\ (\beta)$	$10 - 20\ s$
Pitch sweep	$\pm 5\ deg\ (\theta)$	$< 10\ s$
Bank sweep	$\pm 5\ deg\ (\phi)$	$< 10\ s$
Beta sweep	$\pm 20\ deg\ (\beta)$	$< 10\ s$

Table 5.1: Details of basic training maneuvers

with the aim of collecting information on leveled flight conditions, while the *sweep* maneuvers are designed to excite the short period mode of the airplane. For this goal, and on the basis of the results that emerged from the previous chapter, very light turbulence (which is often encountered by actual aircraft at low altitudes) could be added for very short time (less than 5% of the whole training maneuver) to better excite short period aircraft modes [65]. At the end of the training maneuver, a slow down, from V_{max} to V_s, was introduced to investigate the low velocity regime, with all the possible flap settings. All the flight simulated in this chapter were obtained using the real Sky-Y autopilot modes used during real flights.

The test maneuver set was composed of several high-dynamic maneuvers and mixed of the three aircraft dynamics in order to evaluate the neural network performance



(a) Angel of attack (b) Angle of sideslip

Figure 5.3: Aerodynamic angles obtained during the Sky-Y training maneuvers

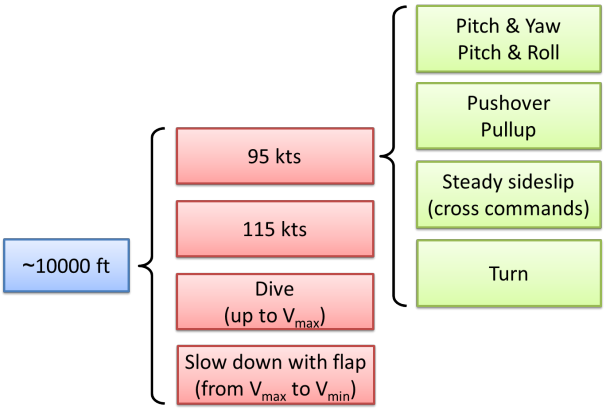
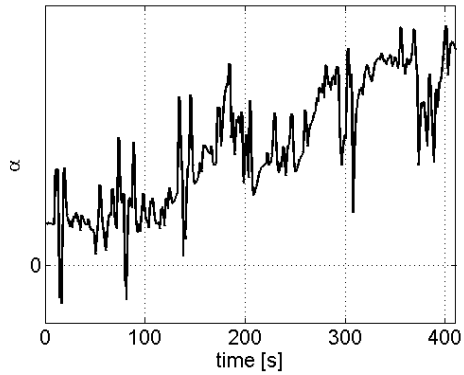


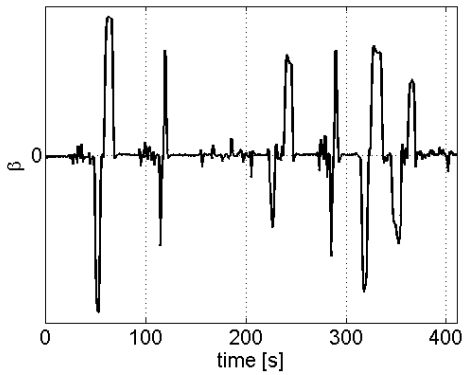
Figure 5.4: Guideline scheme for the test maneuver set

in extreme conditions, as mentioned at the beginning of this section.

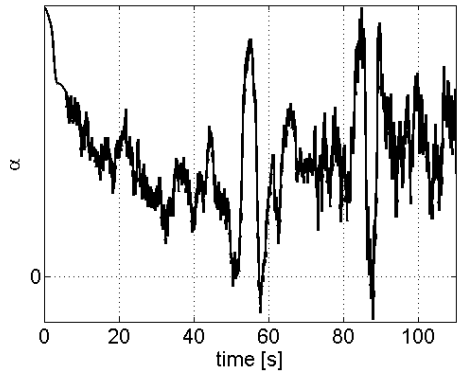
The test maneuver strategy is depicted in Fig. 5.4, where no limits were prescribed to single maneuvers in order to evaluate the real neural network performance. In particular, pitching motion was combined with yawing and rolling maneuvers in order to evaluate the performance of the neural network in estimating superimposed motion that was not performed during the training stage. Pushover and pullup maneuvers were then performed to evaluate the virtual sensor result when very low and very high angles of attack are achieved in unsteady conditions. This group of maneuvers was indicated as *classic*. Steady sideslip, obtained with cross-commands, was performed to obtain high angles of sideslip ($\beta > 10\ deg$) in order to evaluate the extrapolation capability of the neural networks. The dive was used to investigate the neural network performance at higher velocities than those obtainable in realistic



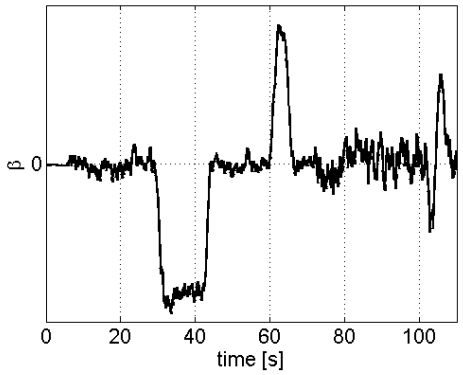
(a) Angel of attack during several simultaneous combinations of yaw, roll and pitch



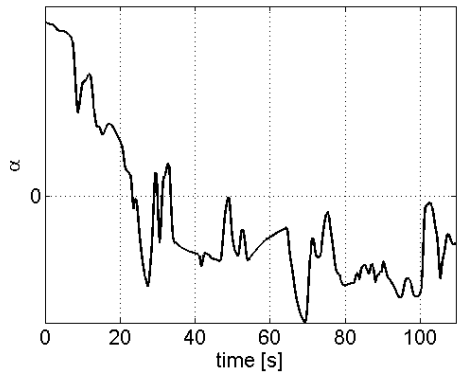
(b) Angle of sideslip during several simultaneous combinations of yaw, roll and pitch



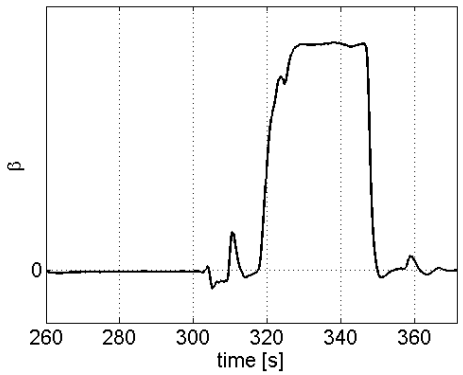
(c) Angel of attack during light and medium turbulence



(d) Angle of sideslip during light and medium turbulence



(e) Angel of attack during dive



(f) Angle of sideslip during steady β with cross commands

Figure 5.5: Aerodynamic angles profiles during different Sky-Y test maneuvers

flight, as was done during the training stage, in other words, the velocity was increased up to V_D of Sky-Y. Again, slow down was performed to evaluate the performance of the virtual sensors when manoeuvring, even with the flaps down. The best strategies have been tuned considering the experience gained with the Beaver simulator. The most important aircraft dynamics observed during test maneuvers are reported in Fig. 5.5.

5.2 Modified Virtual Sensor: New Input Vectors

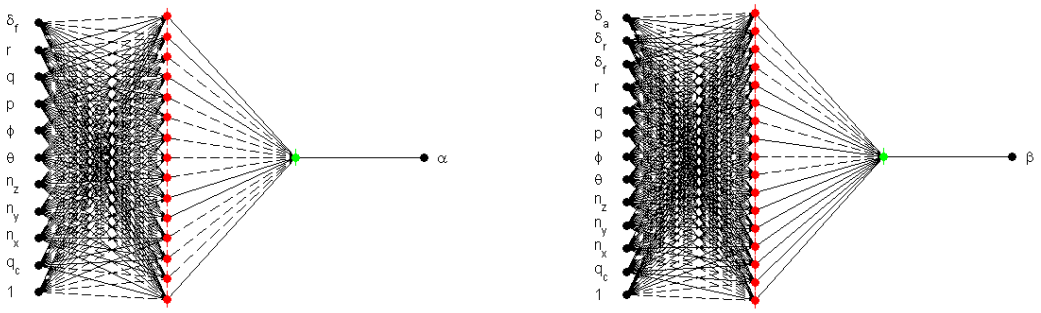
As stated in the previous section, the neural networks designed for Sky-Y were designed to be used during the maneuvers with the flaps down. NNA and NNB designed for Beaver simualtor neede to be modified in order to consider the flap position signal. Two new neural networks were therefore introduced and depicted in Fig. 5.6. The flap deflection signal was used as input with respect to the Beaver application and some of the surface control deflections were removed from the input vectors when was seen that they had no influence on the NN performances at any extent. In fact, several configurations were tested and the results are summarized in Tab. 5.2. In particular, the importance of commands in the input vectors was

variable	inputs	MSE [deg]	Max error [deg]
α	all inputs	$8.2 \cdot 10^{-5}$	0.25
	w/o δ_e	$9.4 \cdot 10^{-5}$	0.27
	no commands	$1.1 \cdot 10^{-4}$	0.27
β	all inputs	$1.0 \cdot 10^{-4}$	0.47
	w/o δ_e	$1.5 \cdot 10^{-4}$	0.53
	w/o δ_r and δ_a	$7.6 \cdot 10^{-3}$	1.7

Table 5.2: Performance of neural networks changing the commands in the input vector

investigated. Some position demands were cut from the input vectors in order to understand their influence on the aerodynamic angle estimations of the neural networks. Tab. 5.2 suggests that the use of commands, δ_e, δ_a and δ_r, could be excluded from the input vector for neural network for angle of attack, α, because it implied a slightly worse performance. Instead, as far as the neural network for angle of sideslip estimation is concerned, acceptable performance were obtained using at

least rudder and aileron deflections as inputs. Therefore, the final version of the neural networks was characterized for α and for β, as depicted in Fig. 5.6, and, since the inputs were increased for NNB, the neuron number was also increased, from 15 to 17.



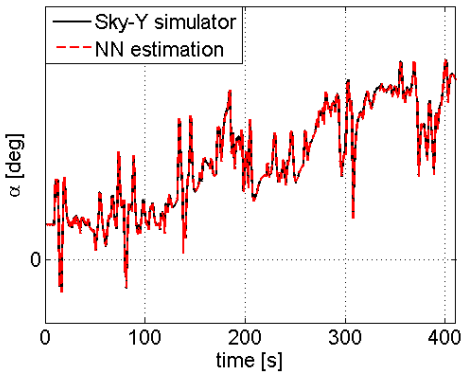
(a) (b)

Figure 5.6: Modified neural networks for the angle of attack (a) and sideslip (b) estimation. The dashed connections represent negative synaptic weight, the solid connections represent positive synaptic weight

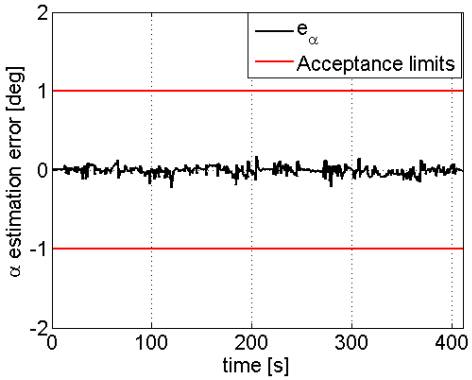
The subsequent neural network performances were obtained using the several test maneuvers described before. As expected, the errors of the aerodynamic angle estimation were very small when maneuvers similar to those of the training set are performed. The real validation of the neural network was carried out using the test maneuvers presented in Fig. 5.5. In the following figures, the absolute values are plotted in the left columns and the absolute errors of the estimation performed by NNs in the right columns both for the angle of attack and sideslip; where errors, defined in (3.12), are

$$e_\alpha = \hat{\alpha} - \alpha_\infty \text{ and } \quad e_\beta = \hat{\beta} - \beta_\infty$$

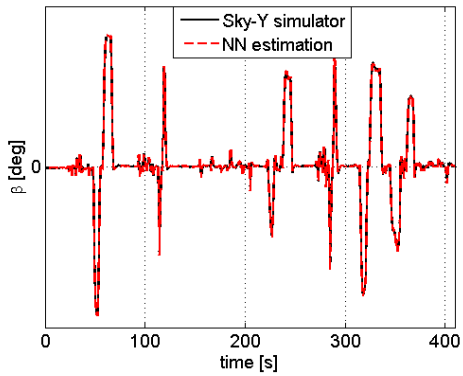
for angle of attack and sideslip, respectively. The errors of the estimation of the angle of attack, during maneuvers, in which there was a simultaneous combinations of yaw, roll and pitch were performed, were very small and contained within ± 0.3 *deg*. Outside the training maneuver boundaries, when in dives the V_{NE} was exceeded, the extrapolation was still good: the estimation error was again within ± 0.3 *deg*, but the most important aspect was that the neural network did not generate diverging



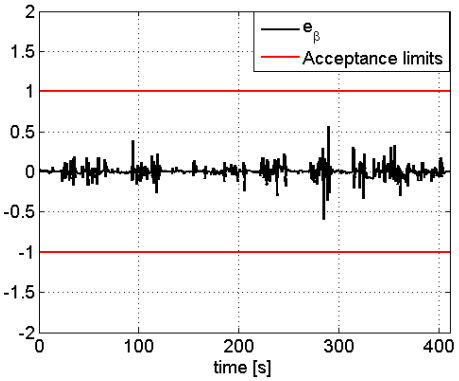
(a) Reference and Estimated angle of attack during several simultaneous combinations of yaw, roll and pitch



(b) Error of angle of attack estimation during several simultaneous combinations of yaw, roll and pitch

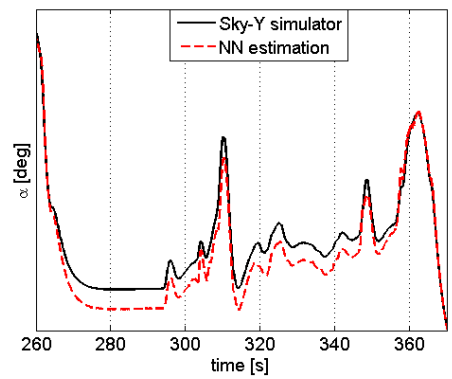


(c) Reference and Estimated angle of sideslip during several simultaneous combinations of yaw, roll and pitch

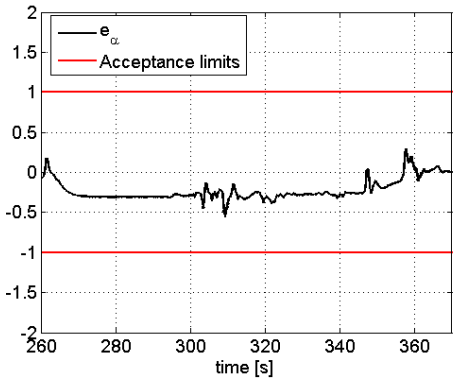


(d) Error of angle of sideslip estimation during several simultaneous combinations of yaw, roll and pitch

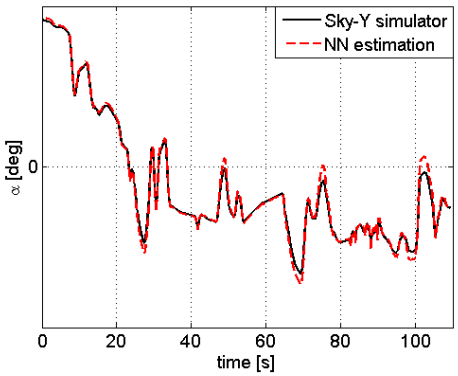
Figure 5.7: Test of the neural network designed for aerodynamic angle estimation using the Sky-Y simulator performing test maneuvers similar to those performed during actual Sky-Y flight tests



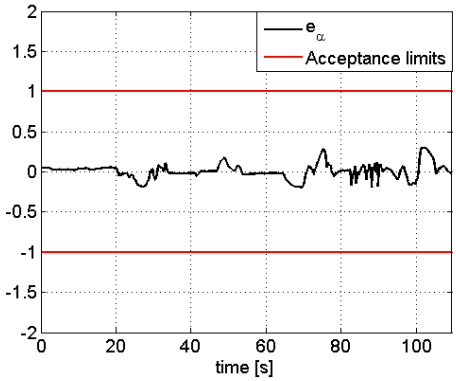
(a) Reference and Estimated angle of attack during steady sideslip



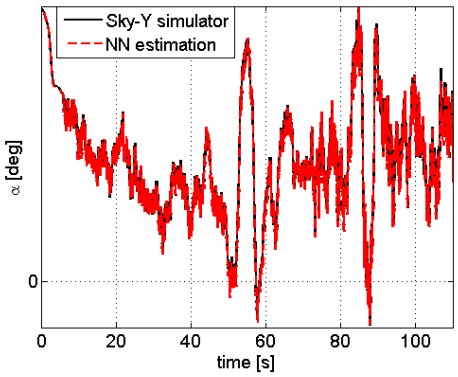
(b) Error of angle of attack estimation during steady sideslip



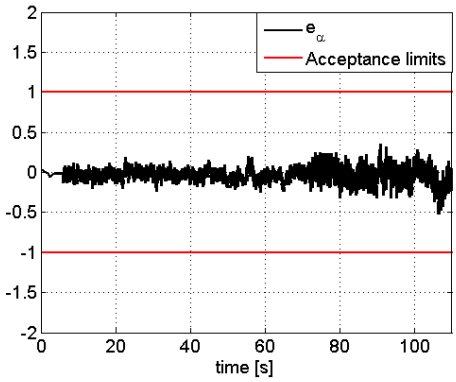
(c) Reference and Estimated angle of attack during dive



(d) Error of angle of attack estimation during dive



(e) Reference and Estimated angle of attack during light and medium turbulence



(f) Error of angle of attack estimation during light and medium turbulence

Figure 5.8: Test of the neural network designed for for angle of attack, α, estimation using the Sky-Y simulator performing the test maneuvers

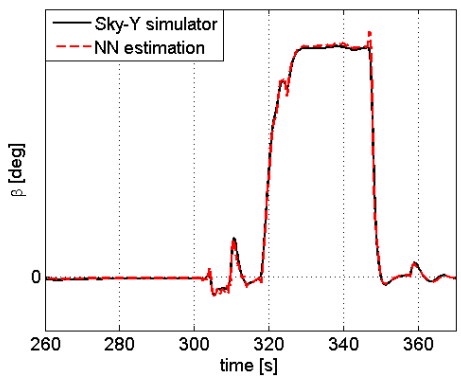
outputs. When turbulence was activated, the maximum absolute errors of the estimation increased to 0.85 *deg*. Results of presented here (Fig. 5.9) are worse than those presented in section 4.4. This is mainly due to accuracy of all calculation processes implemented in the Sky-Y simulator: time dalays due to filtering and the non-synchronization between all the aircraft subsystems were in fact simulated and led to larger estimation errors if compared to those obtained for the Beaver simulator. Overall, the validation of the neural network for angle of attack estimation could be considered positive, since the maximum error is within the acceptance limits and, more important, the neural network works very well even in the presence of high dynamic maneuvers and medium turbulence.

The errors of estimation on angle of sideslip, during maneuvers in which there was a simultaneous combinations of yaw, roll and pitch were simulated, were very small and within ± 0.6 *deg*, but were higher than those found for α. When flying with air turbulence, the absolute errors of the estimation increased to 0.95 *deg* which was comparable with those obtained for the angle of attack. During dives, the extrapolation was almost acceptable: the errors of the NN estimation were within ± 1.1 *deg*. However, dives were simulated with the aim of investigating if the neural network produced diverging outputs, since speeds hgher than V_{NE} were not realistic flight conditions. Overall, the validation of the neural network for the angle

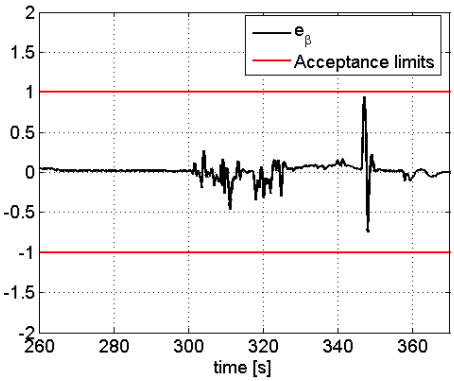
	Type of maneuver	Absolute max error [deg]
α	classic	0.30
	turbulence	0.85
	$V > V_{NE}$	0.30
β	classic	0.60
	turbulence	0.95
	$V > V_{NE}$	1.1

Table 5.3: Summary of virtual sensor performance

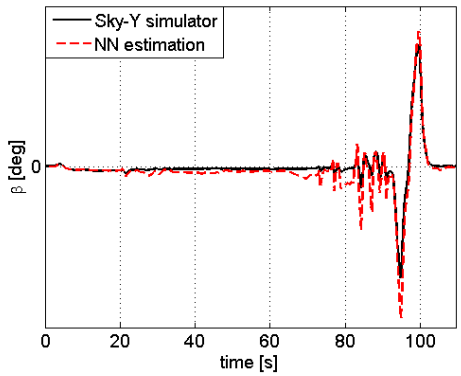
of sideslip estimation, summarized in Tab. 5.3, can be considered positive since the neural network performance was acceptable, even in presence of high-dynamic maneuvers and medium turbulence, although errors beyond the acceptance limits occurred for unrealistic flight conditions.



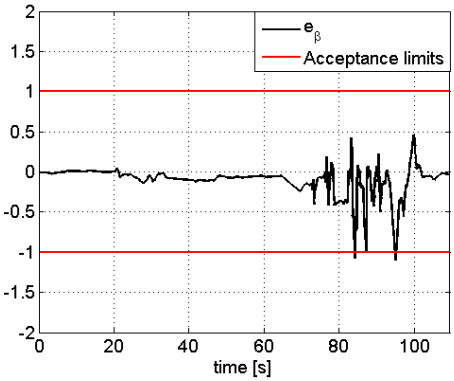
(a) Reference and Estimated angle of sideslip during steady sideslip



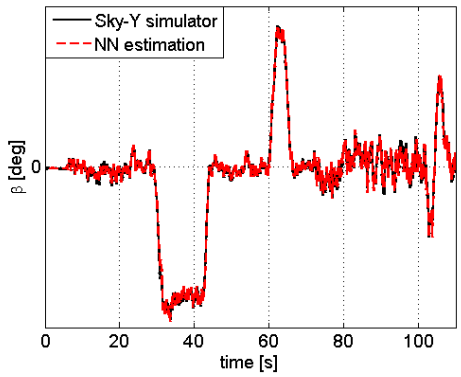
(b) Error of angle of sideslip estimation during steady sideslip



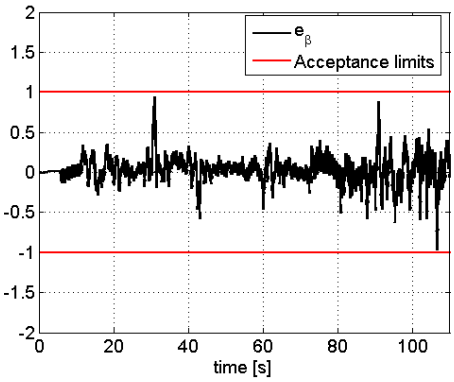
(c) Reference and Estimated angle of sideslip during dive



(d) Error of angle of sideslip estimation during dive



(e) Reference and Estimated angle of sideslip during light and medium turbulence

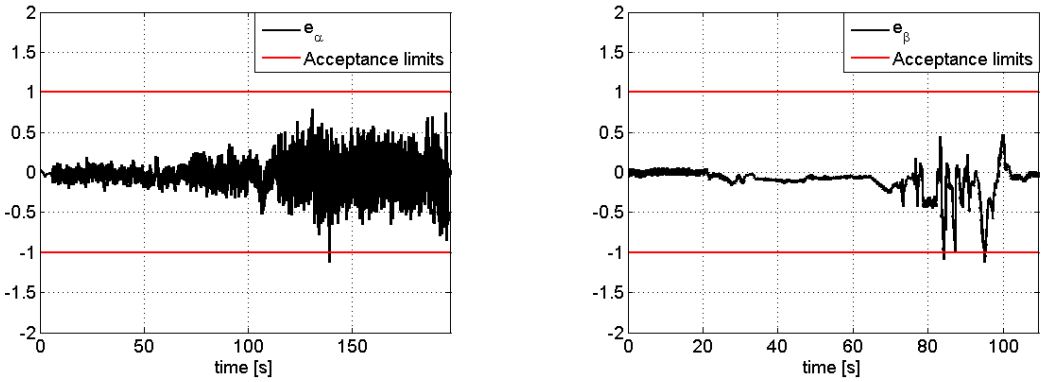


(f) Error of angle of sideslip estimation during light and medium turbulence

Figure 5.9: Test of the neural network designed for angle of sideslip, β, estimation using the Sky-Y simulator performing the test maneuvers

5.2.1 Performance Simulating Sensor Noise

In this section, the neural network were tested in the presence of measurement noise of sensors which provide input signals to NNs, as described in 4.4.3, while the noise derived from structural vibration was discussed in the section 4.4.4. The performance of the virtual sensors were not highly influenced by noise of current pressure transducers and inertial sensors, as desribed in the section 4.4.4. Fig. 5.10, shows the worst case for α (during turbulence) and for β (during diving), which were found in the previous section. Overall, the neural networks used for the angle of



(a) Error of angle of attack estimation during light and medium turbulence

(b) Error of angle of sideslip estimation during dive

Figure 5.10: Virtual sensor performance during the worst possible flight condistions simulating realistic sensor noise

attack and sideslip estimation used for the Sky-Y simulator were validated against severe flight conditions and realistic sensor noise without filtering. Therefore, these network were now ready to be tested on real hardware for a final validation.

5.3 Validation on Real FCC

In this section, the virtual sensors were tested in a real hardware environment with he aim of assessing:

- the integrity of the software (written and compiled in-house),
- the real execution time (t_{exe}),

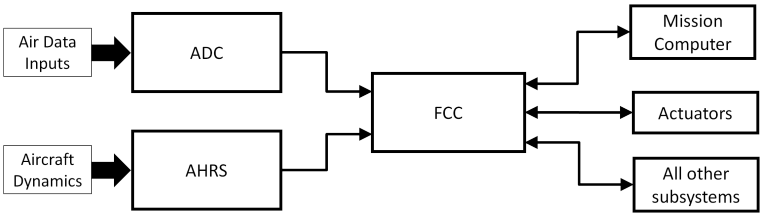


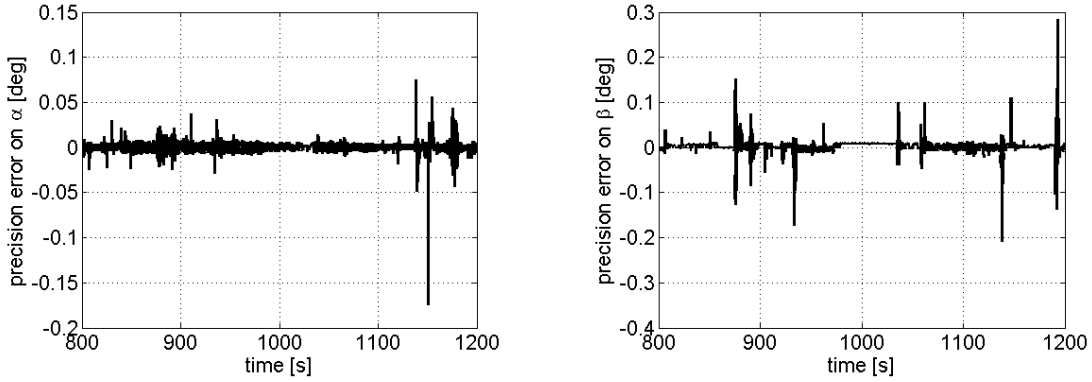
Figure 5.11: A possible of UAV subsystem working scheme. The FCC receives several inputs from ADC and AHRS and communicates with other subsystems

- the estimation performance of aerodynamic angles, α and β.

The present activity included the development of an executable code and its subsequent download into a real flight control computer. This was followed by a laboratory testing session conducted at the Alenia Aermacchi integration rig. This integration rig includes a flight simulator which had the capability of including hardware in the loop simulations, thus allowing the virtual sensors to be tested in a real hardware environment using similar maneuvres to those previously described for the testing phase (see section 6) for a data comparison. The Sky-Y flight simulator and the rig simulator are identical, in terms of dynamic models and operations (e.g. hardware schemes of Fig. 5.11), the only differences being the necessary connections of the real aircraft equipment with the rig test bench and the simulator computer (e.g. speed of computations). These differences have been assessed separately as being of no impact on our simulation tests. As is clear, the virtual sensors need to be translated into a language that is able to be interpreted by the Sky-Y FCC. In this case, all the Matlab routines and subroutines, such as the sigmoid function of the hidden neurons, were previously translated into a C++ code and then adequately compiled in order to be downloaded into the FCC. Although the series of signal manipulation from all the subsystems in the previous Sky-Y simulations were simulated, the calculations speed could only be tested on real hardware. As stated at the beginning of this work, the final neural networks have to run in a few milliseconds: the current virtual sensors need less than half a millisecond ($t_{exe} < 0.5\ ms$), from the input reading to output generation, to estimate aerodynamic angles.

5.3.1 Virtual Sensor Test

A preliminary test was carried out to evaluate the difference in precision between Matlab, which works in double precision (64 *bit*), while the real FCC did not manage operations using numbers greater than 32 *bit*. Moreover the input signals are sampled according to their own processing series. Fig. 5.12 shows the typical gap that occurred between Matlab and real FCC during the aerodynamic angle estimation of the neural networks. Unfortunately, the errors were sometimes quite

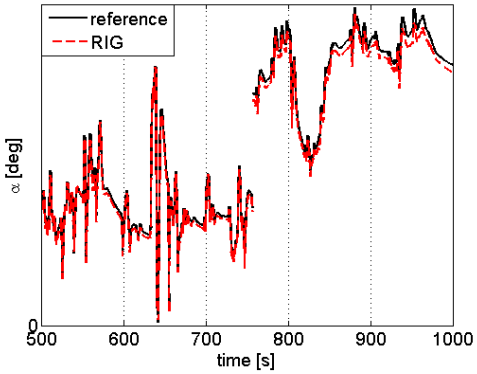


(a) Typical precision error ($\hat{\alpha}_{FCC} - \hat{\alpha}_{Matlab}$) on angle of attack

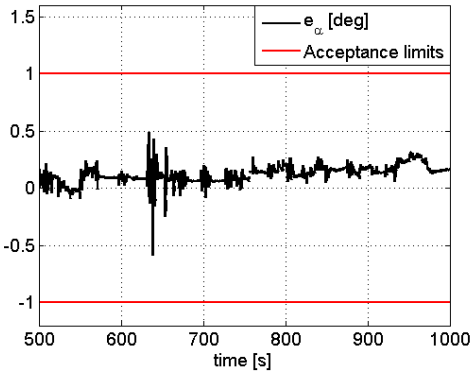
(b) Typical precision error ($\hat{\beta}_{FCC} - \hat{\beta}_{Matlab}$) on angle of sideslip

Figure 5.12: Precision errors of the neural network estiamtion between Matlab and Sky-Y FCC

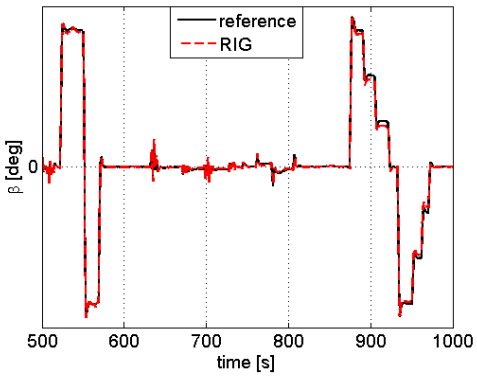
large and there was no possibility of reducing them, because they depended on the hardware characteristics of the used FCC. Therefore, even though simulation with real FCC produced greater errors than those obtained presented in section 5.2, they did not influence the absolute errors of the neural network estimations to any extent. First, a flight simulation (Fig. 5.13), similar to a real Sky-Y flight test, was carried out to assess the virtual sensor performance during a realistic flight, in which light turbulence, low aircraft dynamics and flap deflections are involved. Both the angle of attack and sideslip were estimated by virtual sensors with a very good accuracy, the maximum errors are always within ± 0.8 *deg*. As described in section 6, the virtual sensors were tested during high dynamic test maneuvers with full authority commands, where medium turbulence was also involved, in order to evaluate the performance of the neural network during extreme conditions. The neural networks



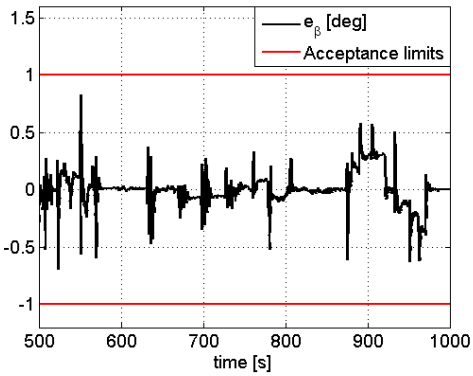
(a) Reference and estimated angles of attack during several simultaneous combinations of yaw, roll and pitch



(b) Error of angle of attack estimation during several simultaneous combinations of yaw, roll and pitch



(c) Reference and estimated angles of sideslip during several simultaneous combinations of yaw, roll and pitch



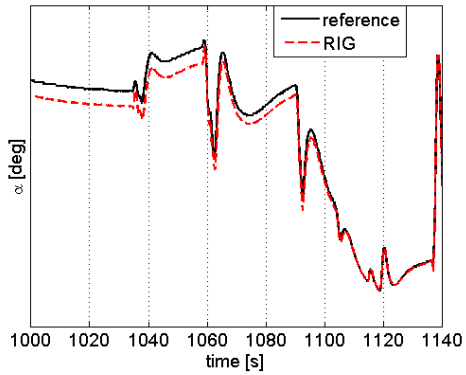
(d) Error of angle of sideslip estimation during several simultaneous combinations of yaw, roll and pitch

Figure 5.13: Validation, on actual FCC, of the neural network designed for aerodynamic angles of attack estimation using test maneuver similar to those performed during actual Sky-Y flight tests

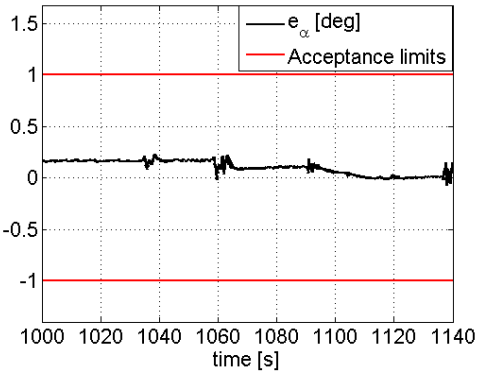
were also tested outside the real Sky-Y flight envelope to investigate the ability of the virtual sensor to generalize and extrapolate on real hardware. Fig.s 5.14 and 5.15 show the performance of the virtual sensors both for angle of attack and sideslip. During the classic maneuvers shown in Fig. 5.14(a), the virtual sensor showed a maximum error (Fig. 5.14(d)) bounded within 0.58 *deg* for angle of attack estimation. During flight with light and medium turbulence levels (Fig. 5.14(e)), higher errors occurred as can be seen in Fig. 5.14(f), where the maximum absolute error is less than ± 1.2 *deg*. However, the performance can be considered acceptable, even in the simulated environmental conditions. The performance of the virtual sensor still remained acceptable even for speeds higher than V_{NE}. The maximum errors (Fig. 5.14(d)) were in fact within ± 0.59 *deg*, and the virtual sensor is as reliable as in normal conditions even in this unrealistic flight conditions. Moreover, it was proven that the angle of attack estimation does not diverge, even when the virtual sensor was working outside the training boundaries or outside the real operating flight envelope of the aircraft. Fig. 5.15 shows that the errors of β estimation during test maneuvers. During steady sideslip (Fig. 5.15(a)), where $\beta > 10$ *deg* were obtained, the maximum error (Fig. 5.15(b)) was within ± 0.98 *deg*. This error lasted for a single time step, and occurred during the maximum slope when the synchronization of the NN estimation with the reference signal was obviously a key factor. Moreover, the points close to the acceptance limits are mainly due to high-dynamic maneuvers, which were achieved using step commands (never used during training stages), during medium turbulence. It was noted that, in the same conditions, piloting the aircraft using a conventional pilot's control stick, the maximum errors were again within the tolerance limits.

Higher errors are shown in Fig. 5.15(f) when maneuvering during light and medium turbulence (Fig. 5.15(e)), these errors could be enclosed in a ± 1.5 *deg* range for medium level and ± 0.7 deg for light level. At this stage must be considered that UAVs do not usually fly in medium turbulence for safety reasons, and the test was therefore needed to evaluate if some problems, such as divergence of outputs, emerged during turbulent flight simulations.

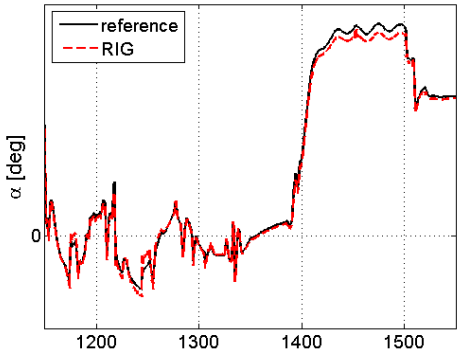
As far as velocities higher than never exceed are concerned, the maximum errors (Fig.



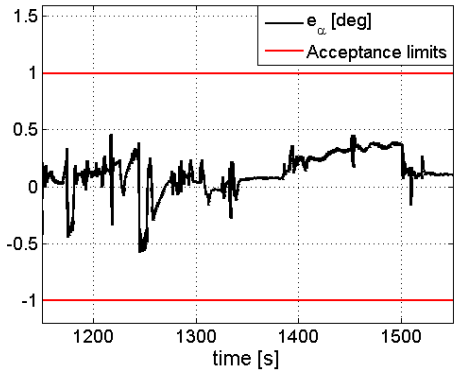
(a) Reference and estimated angles of attack during steady sideslip



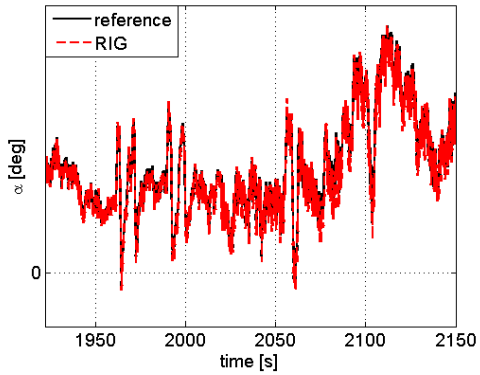
(b) Error of angle of attack estimation during steady sideslip



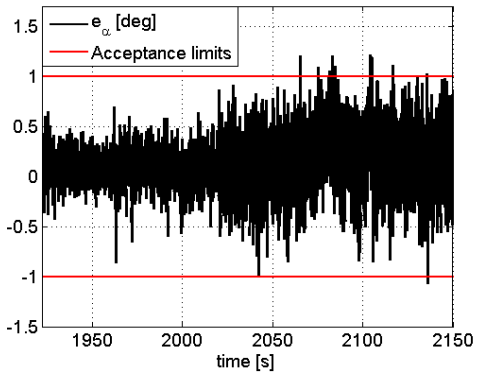
(c) Reference and estimated angle of attack during dive



(d) Error of angle of attack estimation during dive

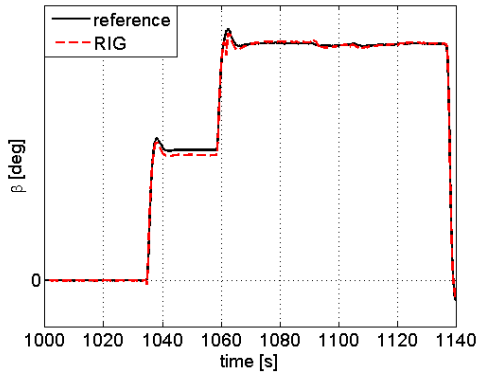


(e) Reference and Estimated angles of attack during light and medium turbulence

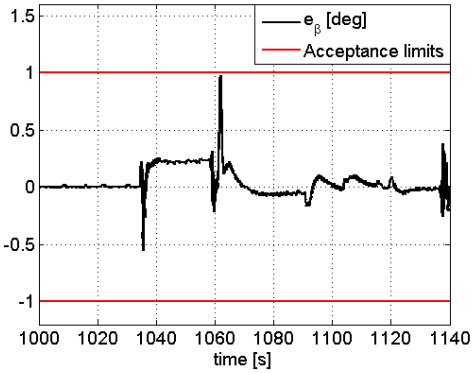


(f) Error of angle of attack estimation during light and medium turbulence

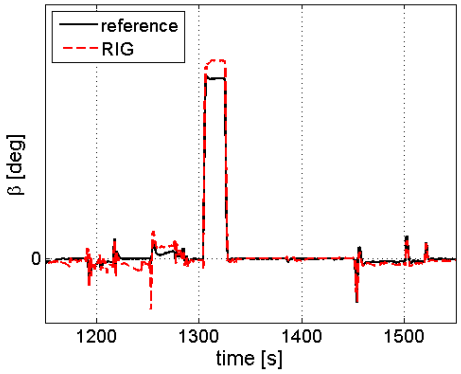
Figure 5.14: Validation on actual FCC of neural network designed for angle of attack, α, estimation using test maneuver. The red lines represents the acceptance limits



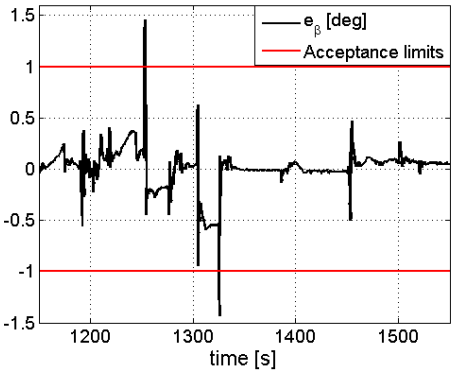
(a) Reference and estimated angles of sideslip during steady sideslip



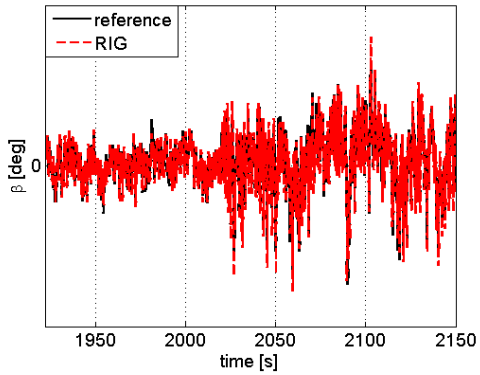
(b) Error of angle of sideslip estimation during steady sideslip



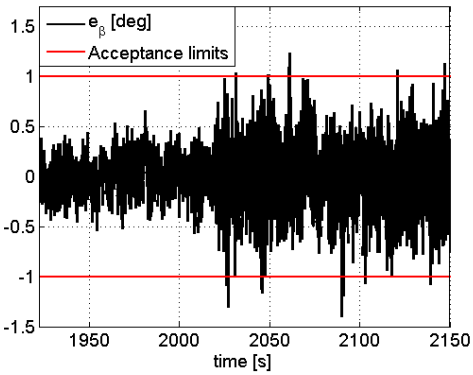
(c) Reference and estimated angles of sideslip during dive



(d) Error of angle of sideslip estimation during dive



(e) Reference and estimated angles of sideslip during light and medium turbulence



(f) Error of angle of sideslip estimation during light and medium turbulence

Figure 5.15: Test on actual FCC of the neural network designed for angle of sideslip, β, estimation using test maneuver. The red lines represent the acceptance limits

5.15(d)) are within ± 1.5 *deg*. Even in this unrealistic flight condition, the virtual sensor is almost a reliable predictor of the angle of sideslip since the estimation did not diverge and the overall accuracy was bounded within ± 1.5 *deg*. A considerations must be made about this flight condition for which the errors of β estimations were not acceptable. Tests were carried out beyond the training boundaries in order to evaluate the neural network performance outside the real aircraft flight envelope and mainly to investigate the occurrence of diverging outputs. Therefore, maximum errors of less than ± 1.5 *deg* confirmed that NNs do not diverge and the NN predictions can be still considered reliable aerodynamic angles estimation even during extreme flight conditions.

Overall, the global performance of the virtual sensor for β estimation can be considered acceptable, considering that a maximum error of the angle of sideslip prediction bounded in ± 1.5 *deg* could also be obtained, during medium turbulence.

In conclusion, although maximum errors are higher than acceptance limits in some extreme situations, such as maneuvers at speeds higher than V_{NE} (up to V_D) or during medium turbulence, the virtual sensors were still suitable aerodynamic angles estimators without diverging. The errors greater than the tolerance limits are only experienced for isolated time steps, due to the high dynamics involved and step commands, but this did not represent a real problem because the errors are bounded. Moreover, this kind of errors can be easily removed, for example, using a high frequency cutting filter. Therefore, the current neural networks designed for angle of attack, α, and sideslip, β, estimation may also be considered ready to be tested in real flight conditions.

5.4 Aeronautical Certification

Over the last few decades, *soft-computing* techniques have reached in last decades such a stage of maturity stage that they can be used in real world applications [74]. These techniques can be characterized by *on-line* and *off-line* training methods, see section **??**. The off-line training strategy was used in this work because it allows neural networks, or other *soft* techniques, to be a deterministic software

after training, like any other software on modern FCCs. On the other hand, the on-line training technique allows neural networks, or any other *soft* techniques, to be trained during their operative lives in order to always be updated and able to adapt to system evolutions: when damage or failures occur, the software can re-adapt to the new and unusual system configuration. Clearly, this kind of training strategy is not deterministic and it is rather hard to certify according to current airworthiness certification regulations. Today, the only training method that can be certified is the off-line strategy which can assure a deterministic software after the training stage. However, even though the off-line approach is used, several aspects still need to be considered before full aeronautical certification. In fact, before an NN can be used in safety critical applications, like UAV intended to fly over populated and not segregated areas, a certification process must be established. In this context, certification is the process of obtaining a certificate from an authority (e.g. ENAC in Italy) to indicate conformance with airborne software standards and aircraft certification specifications (usually DO-178B, CS-23, AER-P.2). As a general overview, the airborne software standards provide guidelines for the production of software for systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. These guidelines are in the form of:

- objectives for software life cycle processes,
- descriptions of activities and design considerations for the achieving of those objectives,
- descriptions of the evidence that indicate that the objectives have been satisfied.

The aircraft certification specifications typically contain a set of technical airworthiness requirements that are primarily intended for the airworthiness certification of manned and unmanned airplanes that are intended for regular use in unsegregated airspace. The certifying authorities may apply these certification requirements outside these limits whenever appropriate.

5.4.1 Life Cycle of the Neural Network Certification

A pre-trained NN, such the one used in this work, is a purely deterministic mapping, and it may be analyzed just like any other function. Hence, the verification should not be more difficult than that of other well accepted implementations of non-linear mappings, such as the polynomials or look-up tables currently employed in air data computers for the storage of PEC and other air data calibration functions. A possible high-level verification and validation process that could be used to ensure that the design of the NNs, or virtual sensors, yield a safe system, in compliance with requirements, is:

- **definition of the system requirements and the software requirements note**. The documentation should include the specification for the NN and its architecture (e.g.: learning algorithms, number of layers, description of inputs and outputs, etc.).
- **Definition of the training data** (the data should consist of all the variables used as input as well as the desired output of the system).
- **Definition of the test data note**. In order to test the NN, it may be necessary to design and develop specific verification tools in parallel, but independent from the primary developers. If possible, the non-diverging neural network outputs should be demonstrated when they exceed the training data set.
- **Code verification**, pertaining to the NN software rather than the training of the NN, in order to ensure that:
  - the code is traceable to the design and the requirements,
  - the code can be derived from the design and requirements,
  - the code implements the safety and other critical requirements correctly
- **Integration verification**. It is assumed that the NN is an integral part of another system. For example, the NN developed for the virtual sensors in this work is an integral part of the flight control computer. Therefore, it

is important to verify that all the inputs/outputs between the systems are properly scaled and that the NN properly interfaces with the FCS.

- **Documentation verification** to ensure that the documentation is adequate, consistent and complete and, moreover, that the configuration management of the documents follows specified procedures.

Chapter 6

Sensitivity Analysis of the Virtual Sensors

Sensitivity analysis is the study of how the uncertainty in the output of a mathematical model or system can be apportioned to different sources of uncertainty in its inputs [75]. In this chapter the sensitivity analysis was applied to virtual sensors based on neural networks in order to evaluate

- the robustness of virtual sensors in response to uncertainty on inputs or simulated failure of sensors which provide inputs;
- identify strong and weak relationships between inputs and output in order to propose simplified neural networks.

In particular, the sensitivity analysis for an aircraft model could be performed in different ways depending on the flight condition characteristics. Steady flight, for example, could be analyzed through a local method involving the linearization of the equations of motions:

$$f(\dot{X}, X, U) = 0, \tag{6.1}$$

where X and U are respectively the state and control variables and f indicated the implicit nonlinear body-axis first order differential equations of motion [65]. Linearization around the steady condition implies calculating the partial derivatives of each equation with respect to each variable:

$$\nabla_{\dot{X}} f \delta\dot{X} + \nabla_X f \delta X + \nabla_U f \delta U = 0, \tag{6.2}$$

where ∇ represents a row vector of first partial derivative operators. Analytical and numerical investigations reveal that, under the specific assumption that the stability-axis inertia matrix J_s [65] is symmetric, the longitudinal and lateral-directional equations are decoupled. This implies that α is affected mainly by the longitudinal variables q_c, n_x, n_z, q, θ and δ_f, while ÌĞβ is affected mainly by the lateral-directional variables q_c, n_y, p, r, ϕ. In unsteady conditions the non linear dynamic equations must be considered and the sensitivity analysis can be performed through the uncertainty propagation method [72, 73], which assumes independence among the measured variable: test maneuvers were simulated in the time-domain and the sensor input signal was modelled assuming that uncertainties had a Gaussian standard probability distribution, where the root mean square deviation was given by the particular sensor accuracy. In fact, the virtual sensors could be considered as a classical measuring devices with their own performance, in terms of accuracy and robustness to external disturbances. The accuracy of measurements is in fact a specific characteristic of the particular sensor, which is declared by manufacturers in datasheets. The uncertainty analysis was performed in unsteady conditions over all the variables included in Eq.s 4.17 in several flight conditions. In real operations, the virtual sensor works with input signals which come from measurements of dedicated probes or sensors with their own uncertainties. Four sources of uncertainties on NN inputs were considered in the present analysis: measurement accuracy of sensors, which provide NNs with input signals, and three failure modes that will be described later on. Moreover, for the sake of generality, in the present analysis, accuracy was given as a percentage deviation with respect to the nominal value inspired by current sensor data-sheets [68, 67, 69, 62] without any reference to the particular sensor.
Results are presented in tables containing the maximum absolute error and MSE of NN predictions with reference to any level of single input corruption: each input was considered to fail individually because no hypothesis were here assumed on the sensor platform which provide neural network with inputs.
The way to read all the tables of this chapter is here described. The first row of all tables indicates the kind of simulated flight. By *classic* we intended maneuvers that were performed during the test stage, which are described in the section . In

approach were grouped those classic maneuvers with speed below $100kts$ in order to simulate the approach and landing phases using all the available flap settings. Finally, *turbulence* is about to indicates that set of classic maneuvers performed during light and medium turbulence, which was simulated according to the Dryden model [66, 76]. All the inputs, for which individual failure were considered, are reported in the first column. The numbers inside the tables are the absolute errors of NN estimation. The second rows indicates the type of failure and its damage degree. Let us consider, for example, to be interested in how the angle of attack estimation degrades during the approach phase when the accuracy of the measured dynamic pressure, q_c, is 10%. Hence, the row corresponding to considered failed input (q_c in this example) in Tab. 6.1(b) must be crossed with the column of considered degree of failure (10%) in the approach table. Therefore, the cross-checked number is the maximum absolute error produced by virtual sensor for the α estimation when the accuracy on measured q_c is 10%.

6.1 Effect of Accuracy Measurements of NN Inputs

In this section, the several accuracy levels of inputs to NNs will be used as source of external disturbance to the virtual sensors. As previously mentioned, the accuracy is here simulated considering a percentage error of the nominal value.

The virtual sensor was in fact fed by several sensors, each of them with its own accuracy of provided measurements. In particular, the accuracy is a specific characteristic of the sensor which is declared by manufacturers in terms of several contributions: sensor non-linearity, hysteresis and non-repeatability, other than temperature effects on the offset and hysteresis themselves. Therefore, each input signal can be corrupted using a realistic accuracy inspired by current datasheets [68, 67, 69, 62] of the sensor at hand.

As said before, the virtual sensors were simulated to work with a single corrupted input signal by using $\pm 2\%$, $\pm 5\%$, $\pm 10\%$ and $\pm 20\%$ accuracy error with reference to nominal conditions indicated with 0%; in the tables are reported the worst conditions.

Classic					
Input	Nominal	±2%	±5%	±10%	±20%
q_c	0.23	0.38	0.64	1.06	1.85
n_x	0.23	0.23	0.23	0.23	0.23
n_y	0.23	0.22	0.21	0.21	0.20
n_z	0.23	0.37	0.73	1.34	2.59
θ	0.23	0.23	0.23	0.23	0.23
ϕ	0.23	0.23	0.23	0.23	0.23
p	0.23	0.23	0.23	0.23	0.23
q	0.23	0.23	0.23	0.23	0.23
r	0.23	0.23	0.23	0.23	0.24
δ_f	0.23	0.23	0.23	0.23	0.23

(a)

Approach					
Input	Nominal	±2%	±5%	±10%	±20%
q_c	0.52	0.76	1.56	2.63	4.11
n_x	0.52	0.55	0.59	0.66	0.79
n_y	0.52	0.53	0.53	0.54	0.55
n_z	0.52	1.16	2.21	4.23	9.59
θ	0.52	0.52	0.51	0.49	0.46
ϕ	0.52	0.52	0.52	0.52	0.53
p	0.52	0.53	0.53	0.54	0.55
q	0.52	0.53	0.53	0.54	0.56
r	0.52	0.53	0.53	0.53	0.54
δ_f	0.52	0.34	0.53	0.91	1.59

(b)

Turbulence					
Input	Nominal	±2%	±5%	±10%	±20%
q_c	0.52	0.64	0.83	1.12	1.87
n_x	0.52	0.52	0.51	0.51	0.51
n_y	0.52	0.52	0.53	0.53	0.54
n_z	0.52	0.63	0.84	1.25	2.61
θ	0.52	0.52	0.52	0.52	0.52
ϕ	0.52	0.52	0.52	0.52	0.52
p	0.52	0.52	0.52	0.52	0.52
q	0.52	0.52	0.52	0.52	0.52
r	0.52	0.52	0.52	0.52	0.51
δ_f	0.52	0.52	0.52	0.52	0.52

(c)

Table 6.1: Failure analysis for angle of attack, α, estimation when the input signal accuracies decrease. The errors represent the maximum e_α $[deg]$

In Tab. 6.1 the maximum absolute errors for the NN estimation of the angle of attack are reported. It can be seen that for the virtual sensor of the angle of attack, α, the dynamic pressure, q_c, and the vertical inertial acceleration, n_z, are the most influencing inputs. Indeed, in Tab. 6.1(a) is clear that when the accuracy of measures is between $\pm 5\%$ and $\pm 10\%$ the error is greater than the acceptance limits (± 1 *deg*), whereas the other inputs can have an accuracy of up to $\pm 20\%$ without affecting the neural network estimation of the α. It can also be seen that for the current flight conditions, whatever the accuracy on measurements of n_x, θ, ϕ, p, q and r, it do not affect the α estimation, and even errors on n_y measurements improve the performance of virtual senor for angle of attack. This particular behaviour is true only for the current flight conditions. The scenario in fact changes during approach maneuvers or turbulent flight conditions. In fact, the maximum allowed accuracy for measurements of q_c and n_z is less than 2%, in order to keep the error of neural network estimation within the acceptance limits. During these flight conditions, emerges that also n_x has an influencing role in the α estimation, even though an accuracy on measurements of up to 20% do not lead the maximum error of α estimation beyond the acceptance limits. The flap angle deflection, δ_f, which considers the real time position of flaps, obviously affect the NN estimation of the α during the approach maneuvers, when accuracy on flap deflections are worse than 10% with reference to true values (Tab. 6.1(b)). Moreover, some inputs, such as n_y, ϕ, p, q and r, can have very large measuring errors without affecting the angle of attack estimation for the Sky-Y application. Overall, considering all the three flight phases, the vital inputs are q_c, n_z, n_x and δ_f for the approach and landing phases. As done for angle of attack, errors of the angle of sideslip estimation due to loss of the accuracy on inputs are reported in Tab. 6.2. It can be seen in Tab. 6.2(a) that the most influencing quantity for angle of sideslip, β, was the lateral acceleration, n_y, and the dynamic pressure, q_c. Indeed, when the accuracy of n_y measurements was worse than $\pm 10\%$, the error on β estimation, e_β, was greater than the acceptance limits (± 1 *deg*). As far as the dynamic pressure is concerned, the allowed accuracy on measurements can be up to $\pm 20\%$, the maximum error was in fact within the tolerance limits ($|e_\beta|_{\max} \leq 0.96$ *deg*) for the β estimation. All the other inputs do

Classic					
Input	Nominal	±2%	±5%	±10%	±20%
q_c	0.59	0.50	0.46	0.60	0.93
n_x	0.59	0.58	0.57	0.55	0.51
n_y	0.59	0.67	0.81	1.03	1.50
n_z	0.59	0.59	0.59	0.60	0.60
θ	0.59	0.59	0.59	0.60	0.62
ϕ	0.59	0.58	0.58	0.57	0.56
p	0.59	0.58	0.57	0.55	0.53
q	0.59	0.59	0.59	0.59	0.59
r	0.59	0.59	0.60	0.62	0.65
δ_f	0.59	0.59	0.59	0.59	0.59
δ_r	0.59	0.60	0.62	0.65	0.70
δ_a	0.59	0.59	0.61	0.63	0.67

(a)

Approach					
Input	Nominal	±2%	±5%	±10%	±20%
q_c	0.76	0.76	0.76	0.76	0.76
n_x	0.76	0.77	0.77	0.78	0.80
n_y	0.76	0.78	0.81	1.06	1.58
n_z	0.76	0.76	0.75	0.75	0.75
θ	0.76	0.76	0.76	0.76	0.77
ϕ	0.76	0.76	0.76	0.76	0.76
p	0.76	0.76	0.76	0.76	0.76
q	0.76	0.76	0.76	0.76	0.76
r	0.76	0.76	0.75	0.73	0.74
δ_f	0.76	0.77	0.79	0.81	0.86
δ_r	0.76	0.77	0.77	0.77	0.87
δ_a	0.76	0.76	0.76	0.76	0.75

(b)

Turbulence					
Input	Nominal	±2%	±5%	±10%	±20%
q_c	0.96	1.04	1.17	1.36	1.66
n_x	0.96	0.96	0.96	0.96	0.97
n_y	0.96	0.90	0.88	1.25	2.09
n_z	0.96	0.96	0.97	0.99	1.04
θ	0.96	0.96	0.96	0.95	0.95
ϕ	0.96	0.95	0.95	0.95	0.95
p	0.96	0.95	0.96	0.98	1.02
q	0.96	0.96	0.96	0.95	0.95
r	0.96	0.97	0.99	1.04	1.14
δ_f	0.96	0.96	0.96	0.96	0.96
δ_r	0.96	0.95	0.93	0.91	0.91
δ_a	0.96	0.96	0.96	0.96	0.96

(c)

Table 6.2: Failure analysis for angle of sideslip, β, estimation when the input signal accuracies decrease. The errors represent the maximum e_β $[deg]$

not influence significantly the estimation of the angle of sideslip, and, in particular, the higher errors on measurements of n_x, ϕ or p the better estimation of β. This particular behaviour is related to the particular conditions flown during classic maneuver set. The same particular behaviour was not found again during the approach and turbulence maneuver sets. It is clear that the most influencing inputs were again n_y and q_c even for during the approach phase and the turbulent flights, while all other inputs slightly influence the errors on β estimation of NNs.

From the analysis of sensitivity of NN-based β estimator, presented in Tab. 6.2, it is clear that for this specific application some inputs, such as n_z, θ, q δ_a and δ_r, may have very bad accuracies (up to 20%) without affecting significantly the β estimation.

Overall, for Sky-Y application, the neural network α estimation is mainly influenced by accuracy of measured q_c, n_z and δ_f, whereas the β estimation is affected to great extent by accuracy on q_c, n_y, n_z, p and r signals, according to results presented in this section.

6.2 Failure Analysis

In this section, the following three main failure modes of the sensors providing inputs to the NNs are considered

- locked signals
- offset drift
- null signal.

When a sensor stops working properly, the hypothesis of locked signal is realistic, while the null signal is only realistic when the failure is recognized. The offset drift is a another source of error that is well known by manufacturers which can be due to hysteresis, temperature changes or simply to long term stability characteristics. For these reasons, the offset drift, which always occurs in the current sensors during their operative life, cannot strictly speaking be considered a real failure mode but rather just the way the sensor works. However, the offset deviation was here indicated as a failure because deviates the measurements from their true values.

6.2.1 Locked Signals

Locked signals were considered here as one of the expected failure modes of neural network inputs. The input signals to NNs were individually corrupted simply locking the signal at take-off (CAS$\approx$ 70 kts) and at maximum speed (CAS$>$ 140 kts) in order to evaluate the worst conditions for this particular failure mode. The

	Classic			Approach			Turbulence		
Input	Nominal	From TO	From V_{NE}	Nominal	From TO	From V_{NE}	Nominal	From TO	From V_{NE}
q_c	0.23	51.10	5.14	0.52	16.13	8.16	0.52	62.75	4.33
n_x	0.23	0.94	0.94	0.52	2.29	2.29	0.52	0.98	0.98
n_y	0.23	0.27	0.27	0.52	0.42	0.42	0.52	0.49	0.49
n_z	0.23	8.89	8.94	0.52	10.76	10.81	0.52	8.91	8.94
θ	0.23	0.26	0.25	0.52	0.44	0.89	0.52	0.50	0.53
ϕ	0.23	0.23	0.23	0.52	0.52	0.52	0.52	0.52	0.52
p	0.23	0.47	0.47	0.52	0.47	0.47	0.52	0.64	0.64
q	0.23	0.23	0.23	0.52	0.41	0.41	0.52	0.53	0.53
r	0.23	0.22	0.22	0.52	0.46	0.46	0.52	0.53	0.53
δ_f	0.23	2.58	0.23	0.52	11.50	20.85	0.52	2.43	0.52

Table 6.3: Failure analysis for angle of attack, α, estimation when signal lock occurs. The errors represent the maximum e_α $[deg]$

maximum absolute errors of α estimation are reported in Tab. 6.3. Considering the three kinds of maneuvers discussed before, the locked dynamic pressure, q_c, vertical

inertial acceleration, n_z, and flap deflection, δ_f, are those that had most influence on the inputs and led the absolute error of α estimation to go beyond the tolerance limits, while the inertial axial acceleration, n_x, can only cause estimation errors greater than ± 1 *deg* during the Sky-Y approach phase. Instead, no one of the other quantities, when locked during take-off or at maximum velocity, influence the angle of attack estimation carried out by the neural networks for the Sky-Y application to any great extent.

The degradation analysis on β estimation, when the input signals are locked independently, is reported in Tab. 6.4. The input quantities that led the estimation to go beyond

	Classic			Approach			Turbulence		
Input	Nominal	From TO	From V_{NE}	Nominal	From TO	From V_{NE}	Nominal	From TO	From V_{NE}
q_c	0.59	8.54	2.31	0.76	0.76	3.74	0.96	8.53	2.18
n_x	0.59	0.95	0.95	0.76	1.06	1.06	0.96	0.91	0.91
n_y	0.59	6.34	6.34	0.76	4.57	4.57	0.96	8.12	8.12
n_z	0.59	0.95	0.96	0.76	1.85	1.86	0.96	1.33	1.33
θ	0.59	0.64	0.52	0.76	0.81	0.87	0.96	1.01	0.95
ϕ	0.59	1.03	1.03	0.76	1.06	1.06	0.96	1.61	1.61
p	0.59	0.98	0.98	0.76	0.88	0.88	0.96	0.90	0.90
q	0.59	0.57	0.57	0.76	0.76	0.76	0.96	1.01	1.01
r	0.59	1.00	1.00	0.76	1.19	1.19	0.96	1.28	1.28
δ_f	0.59	0.86	0.59	0.76	0.84	1.08	0.96	0.96	0.96
δ_r	0.59	1.30	1.30	0.76	1.50	1.50	0.96	1.87	1.87
δ_a	0.59	0.76	0.76	0.76	0.82	0.82	0.96	1.01	1.01

Table 6.4: Failure analysis for angle of sideslip, β, estimation when signal lock occurs. The errors represent the maximum e_β $[deg]$

the acceptance limits were: dynamic pressure, q_c, lateral acceleration, n_y, bank angle, ϕ, yaw rate, r and rudder deflections, δ_r. However, even if the loss of the ϕ, r and δ_r signals did not have a significant effect on the absolute errors of β estimation (since $|e_\beta|_{\max} \leq 1.30$), n_y and q_c could cause errors grater than 6 *deg* and 10 *deg*, respectively. When the Sky-Y flight was simulated in landing and approach configurations, the aforementioned inputs and also n_z had more influence on error of β estimation. When considering flight in turbulent air, the aileron deflection signal made the angle of sideslip estimation error went outside the acceptance limits, even though the additional error was less than one-tenth degree, the maximum expected error due to locked δ_a was higher than tolerance lmits ($|e_\beta|_{\max} < 1.1$).

From this kind of analysis, it is clear that the most important parameters for an acceptable β estimation were the dynamic pressure, q_c, the three inertial accelerations, the bank angle, ϕ, pitch and yaw rate, q and r, and rudder deflections, δ_r. The flap, δ_f, could be neglected because the maximum absolute errors slightly exceed the acceptance limits, $|(e_\beta)_{\delta_f=0}|_{\max} = 1.08$. Conversely, the pitch angle and roll rate,

θ and p, may be locked for this particular analysis, since their precision was not crucial for neural network β estimation.

6.2.2 Offset Drift

Another failure mode expected in the current sensors that provide NNs with the necessary inputs is the offset drift they experience during their operative lives. Manufacturers usually declare the sensor drift as sum of several factors (temperature effects, hysteresis and long term stability) with reference to the full scale (FS) range of the sensor itself. The FSs of the NN input signal were here inspired by the real probes and sensors used on board the Sky-Y UAV; the following FS values were therefore considered:

- 50 $mBar$ for q_c
- ± 10 g for n_x, n_y and n_z
- ± 90 deg for θ
- ± 180 deg for ϕ
- ± 90 deg/s for p, q and r
- ± 80 deg for δ_r and δ_a.

The results of the present analysis are reported in Tab.s 6.5 and 6.6 for α and β, respectively.

From Tab. 6.5(a), it is clear that when the absolute offset drift is $\leq 0.5\%$ the absolute errors of α estimation are within the tolerance range. If the Sky-Y approach configuration is considered, unacceptable errors already occur when the offset error is only 0.5% for n_x, n_z and q_c; the absolute errors of α estimation are in fact higher than ± 1 deg. Instead, always the offset error is within $\pm 10\%$, none of the other inputs influence the α estimation to any great extent, except for n_y and δ_f whose drift should be less than 5% to avoid exceeding the prescribed tolerance boundaries. Simulating turbulent flights, the scenario was still the same: absolute offset errors $< 0.5\%$ could assure acceptable accuracy of the α virtual sensor, while, as soon as the input offset drift was increased, the q_c, n_x or n_z accelerations can cause estimation errors greater than ± 1 deg. As seen before, there are some inputs, θ, ϕ and p, for which high offset accuracy is not requested for acceptable α estimation for Sky-Y

applications.

As far as angle of sideslip is concerned, Tab. 6.6(a) shows that the most accurate parameters should be lateral acceleration n_y. As the offset drift of n_y is 0.5% FS, the maximum absolute errors of β estimation are in fact greater than three degrees, while for all other inputs it can be noted that as long as the offset deviation is lower than 1% FS, the errors in β estimation are still acceptable, except for n_x for which small errors are produced, $|e_\beta|_{\max} = 1.04$. Almost the same situation is found when the landing and approach configurations are considered. In fact, all the inputs can have drift up to $\pm 1\%$ FS offset, except for n_y, n_x and r, for which a better accuracy is required to assure the acceptability of the neural network β estimations. The scenario becomes worse if turbulence is introduced during the Sky-Y simulations. When the offset drift of $\pm 1\%$ occur, most of the inputs cause large errors on β prediction: all the inputs cause errors greater than 0.95 *deg*. In short, in the analysis of the effects of offset drift on β estimation, all the input quantities must have a better offset drift than half a percent of the corresponding FS (drift$<$ 0.5% FS) in order to keep the accuracy of the neural networks within the tolerance limits.

Classic						
Input	Nominal	±0.50%	±1.0%	±2.0%	±5.0%	±10%
q_c	0.23	0.35	0.51	0.87	1.84	3.07
n_x	0.23	0.70	1.51	6.18	35.87	37.98
n_y	0.23	0.30	0.42	0.85	4.32	22.74
n_z	0.23	0.68	1.30	2.57	7.35	18.62
θ	0.23	0.22	0.22	0.22	0.24	0.29
ϕ	0.23	0.22	0.22	0.21	0.22	0.22
p	0.23	0.24	0.26	0.31	0.47	0.60
q	0.23	0.22	0.22	0.20	0.27	0.46
r	0.23	0.25	0.26	0.27	0.37	0.58
δ_f	0.23	0.32	0.44	0.71	1.40	2.36

(a)

Approach						
Input	Nominal	±0.50%	±1.0%	±2.0%	±5.0%	±10%
q_c	0.52	1.21	2.07	3.36	5.62	7.25
n_x	0.52	1.29	4.33	17.73	37.27	39.93
n_y	0.52	0.63	1.36	3.44	14.78	26.71
n_z	0.52	2.27	4.36	9.94	37.52	70.45
θ	0.52	0.50	0.47	0.42	0.29	0.43
ϕ	0.52	0.53	0.53	0.53	0.53	0.54
p	0.52	0.49	0.46	0.38	0.44	0.64
q	0.52	0.40	0.30	0.46	0.84	1.22
r	0.52	0.57	0.62	0.72	1.02	1.51
δ_f	0.52	0.38	0.47	0.73	1.54	2.72

(b)

Turbulence						
Input	Nominal	±0.50%	±1.0%	±2.0%	±5.0%	±10%
q_c	0.52	0.60	0.69	0.85	1.53	2.61
n_x	0.52	1.09	1.82	8.91	36.32	36.66
n_y	0.52	0.56	0.68	0.76	3.83	21.95
n_z	0.52	0.84	1.16	2.19	6.42	16.23
θ	0.52	0.52	0.51	0.51	0.50	0.52
ϕ	0.52	0.52	0.52	0.54	0.54	0.59
p	0.52	0.53	0.57	0.63	0.69	0.88
q	0.52	0.51	0.51	0.54	0.54	0.68
r	0.52	0.54	0.58	0.61	0.64	0.80
δ_f	0.52	0.60	0.68	0.84	1.30	2.24

(c)

Table 6.5: Failure analysis for angle of attack, α, estimation when input signal offset occurs. The errors represent the maximum e_α $[deg]$

Classic						
Input	Nominal	±0.50%	±1.0%	±2.0%	±5.0%	±10%
q_c	0.59	0.51	0.45	0.53	0.84	1.44
n_x	0.59	0.85	1.04	1.37	2.94	6.32
n_y	0.59	3.26	6.12	11.80	26.89	33.59
n_z	0.59	0.59	0.60	0.60	0.72	0.82
θ	0.59	0.60	0.61	0.64	0.68	0.67
ϕ	0.59	0.62	0.69	0.80	1.04	1.27
p	0.59	0.81	0.97	1.11	1.14	1.10
q	0.59	0.56	0.55	0.53	0.85	1.33
r	0.59	0.68	0.75	1.18	1.97	2.83
δ_f	0.59	0.59	0.61	0.65	0.71	0.80
δ_r	0.59	0.62	0.68	0.79	1.14	1.99
δ_a	0.59	0.66	0.73	0.79	0.88	0.98

(a)

Approach						
Input	Nominal	Null	Nominal	Null	Nominal	Null
q_c	0.76	0.76	0.76	0.76	1.01	1.72
n_x	0.76	0.92	1.19	1.66	1.97	1.59
n_y	0.76	4.55	8.64	15.96	29.46	32.93
n_z	0.76	0.75	0.75	0.75	0.89	1.16
θ	0.76	0.76	0.77	0.78	0.84	0.89
ϕ	0.76	0.73	0.70	0.76	0.92	1.03
p	0.76	0.77	0.80	0.90	1.39	2.07
q	0.76	0.76	0.77	0.81	1.01	1.04
r	0.76	0.97	1.04	1.41	2.33	2.88
δ_f	0.76	0.77	0.78	0.79	0.83	0.90
δ_r	0.76	0.70	0.73	0.85	1.21	1.74
δ_a	0.76	0.78	0.80	0.85	1.04	1.24

(b)

Turbulence						
Input	Nominal	Null	Nominal	Null	Nominal	Null
q_c	0.96	1.01	1.06	1.17	1.44	1.87
n_x	0.96	0.91	1.17	1.71	3.91	7.72
n_y	0.96	2.96	5.42	11.10	24.38	35.88
n_z	0.96	0.97	0.99	1.05	1.09	1.16
θ	0.96	0.95	0.95	0.95	0.98	1.11
ϕ	0.96	0.96	1.02	1.11	1.27	1.40
p	0.96	0.95	0.95	1.04	1.16	1.25
q	0.96	0.98	1.02	1.11	1.33	1.79
r	0.96	1.08	1.31	1.80	2.48	3.21
δ_f	0.96	0.96	0.97	0.99	1.04	1.02
δ_r	0.96	0.99	1.03	1.16	1.56	1.97
δ_a	0.96	0.98	1.06	1.13	1.23	1.13

(c)

Table 6.6: Failure analysis for angle of sideslip, β, estimation when input signal offset occurs. The errors represent the maximum e_β $[deg]$

6.2.3 Null Inputs

In this section, the virtual sensors were supposed to work with independent null input signals and the resulting absolute errors of the neural network estimation were presented. The input signals to NNs were individually set to zero in order to simulate this particular failure mode. As aforementioned, this event may occur when a failure, after being identified by using dedicated monitoring processes, is set to zero by the FCC. The maximum errors obtained during the event of a single null signal are reported in Tab.s 6.7 and 6.8 for angle of attack and sideslip prediction, respectively. During maneuvers representing real Sky-Y flight tests, the parameters

	Classic		Approach		Turbulence	
Input	Nominal	Null	Nominal	Null	Nominal	Null
q_c	0,23	90,72	0,52	85,54	0,52	90,13
n_x	0,23	0,94	0,52	2,29	0,52	0,98
n_y	0,23	0,27	0,52	0,42	0,52	0,49
n_z	0,23	8,45	0,52	10,28	0,52	8,58
θ	0,23	0,25	0,52	0,89	0,52	0,53
ϕ	0,23	0,23	0,52	0,52	0,52	0,52
p	0,23	0,47	0,52	0,47	0,52	0,64
q	0,23	0,23	0,52	0,41	0,52	0,53
r	0,23	0,22	0,52	0,46	0,52	0,53
δ_f	0,23	0,23	0,52	20,85	0,52	0,52

Table 6.7: Failure analysis for angle of attack, α, estimation when null input signals occur. The errors represent the maximum e_α $[deg]$

that led the estimation to go beyond the acceptance limits are: dynamic pressure, q_c, and vertical acceleration, n_z; these inputs, when null, caused errors greater than 8 deg. The lack of all the other inputs did not make the absolute errors of NN estimation, e_α, worse than the tolerance band. During the Sky-Y flight simulations in approach and landing configurations, the axial acceleration, n_x, flap setting, δ_f, and the aforementioned q_c and n_z had the most influence on error of α prediction. In this case, the maximum absolute error was higher than 2 deg, whereas all the other failed signals did not have a significant influence on the neural network estimation of the angle of attack. When turbulent flights were simulated, a similar situation to *classic* maneuvers is reported. In fact, the only two quantities of influence were again q_c and n_z, and these caused errors greater than 8 deg. It was clear, from

this kind of analysis, that the dynamic pressure, q_c, the vertical and axial inertial accelerations, n_z and n_x, and the flap deflections, δ_f, were the key-factors for a successful NN-based α estimator (according to current acceptance limits of ± 1 *deg*). There were some inputs, n_y, ϕ, p, q and r, which did not affect the α estimation at any extent. The maximum absolute errors of β estimation are reported in Tab.

	Classic		Approach		Turbulence	
Input	Nominal	Null	Nominal	Null	Nominal	Null
q_c	0,59	10,39	0,76	2,50	0,96	10,35
n_x	0,59	0,95	0,76	1,06	0,96	0,91
n_y	0,59	6,34	0,76	4,57	0,96	8,12
n_z	0,59	0,91	0,76	1,72	0,96	1,27
θ	0,59	0,52	0,76	0,87	0,96	0,95
ϕ	0,59	1,03	0,76	1,06	0,96	1,61
p	0,59	0,98	0,76	0,88	0,96	0,90
q	0,59	0,57	0,76	0,76	0,96	1,01
r	0,59	1,00	0,76	1,19	0,96	1,28
δ_f	0,59	0,59	0,76	1,08	0,96	0,96
δ_r	0,59	1,30	0,76	1,50	0,96	1,87
δ_a	0,59	0,76	0,76	0,82	0,96	1,01

Table 6.8: Failure analysis for angle of sideslip, β, estimation when null input signals occur. The errors represent the maximum e_β [*deg*]

6.8 when null input signals occur. Considering the maneuvers representing realistic Sky-Y flights, the parameters that led β estimation to go beyond the acceptance limits were: dynamic pressure, q_c, lateral acceleration, n_y, bank angle, ϕ, yaw rate, r, and rudder position, δ_r. However, even though the loss of ϕ, r and δ_r signals did not effect the absolute errors of β estimation to any extent (since $|e_\beta|_{\max} \leq 1.30$), n_y and q_c could cause errors larger than 6 *deg* and 10 *deg*, respectively. When Sky-Y was in landing and approach configurations, n_z, n_x and all the aforementioned inputs had more influence on the errors of β estimation: in this case, the maximum error was less than 5 *deg*. When simulating Sky-Y flights in turbulent air, in addition to the aforementioned inputs (q_c, n_x, n_y, n_z, ϕ, r and δ_r), the aileron position signal, δ_a, and the pitch rate, q could also lead the error of angle of sideslip estimation outside the acceptance limits, even though the expected errors, due to δ_a and q, were very small ($|e_\beta|_{\max} = 1.01$).

Overall, all the inputs of the neural network designed for β estimation were important

for an acceptable β prediction except for pitch angle, θ, and roll rate, p, which may be also neglected according to present analysis, because their presence among the inputs was not crucial for the virtual sensor used as β estimator.

6.3 Considerations

Results presented in this chapter can drive through a deeper understanding of relationships between input and output pattern of present NN.
Results of the sections 6.1, 6.2.3 can give information about the more influencing inputs for NN estimation of aerodynamic angles. As far as α is concerned, relationships between q_c, n_x and n_z and angle of attack prediction were the most important (see Tab. 6.7; the δ_f is vital only when flaps are used. Secondly, even the pitch angle had a weaker influence on the accuracy of the angle of attack estimation (see Tab. 6.7) than aforementioned inputs. Results presented in Tab.s 6.7,6.1 suggest that eliminating all the other inputs (n_y, ϕ, p, q and r) should not affect the α estimation to any extent according to present study. Moreover, as far as n_y, ϕ, p, q and r is concerned, results of the sections 6.2.1, 6.2.2 indicate that as soon as there was a drift or the a signal was locked the errors of α estimation were affected at a great extent. The presence of n_y, ϕ, p, q and r in the input vector of NNA, did not produce any benefit, and as long as there was an error on one of those signals, the NN performance decayed according to the present results. So that, results of the previous section seemed to suggest to build a new input vector for α estimation with q_c, n_x, n_z, θ and δ_f, and the (4.17a) could therefore be reduced as follows

$$\alpha = f_{\alpha,red}\left(q_c,\ n_x,\ n_z,\ \delta_f\right).$$

Results of the sections 6.1, 6.2.3 can give information about the more influencing inputs for NN estimation of aerodynamic angles. As far as β is concerned, relationships between q_c, n_x, δ_r and angle of sideslip prediction were the most important (see Tab. 6.8; the δ_f is important only when flaps are used, but is not as important as for α estimation. Secondly, even n_z, ϕ, p, r had a weaker influence on the accuracy of the angle of attack estimation (see Tab. 6.8) than aforementioned inputs. Results presented in Tab.s 6.8,6.2 seemed to suggest that eliminating all the other inputs

(θ, q, and δ_a) should not affect the β estimation to any extent according to present study. Moreover, as far as θ, q, and δ_a is concerned, results of the sections 6.2.1, 6.2.2 indicate that as soon as there was a drift or the a signal was locked small errors on β estimation were introduced. The presence of θ, q, and δ_a in the input vector of NNB, did not produce any benefits or drawbacks according to the present results. So that, results of the previous section seemed to suggest to build a new input vector for α estimation with q_c, n_x, n_z, θ and δ_f, and the (4.17b) could therefore be reduced as follows

$$\beta = f_{\beta,red}\left(q_c,\ n_x,\ n_y,\ \phi,\ p,\ r,\ \delta_r,\ \delta_f\right).$$

A complete study of proposed new input vectors required a complete analysis about the architecture, as discussed in the section 4.3. At this stage of the research activity, we chose to maintain all the input variables in order to keep the flexibility to modify the neural network working in agreement to particular flight conditions (e.g. the aircraft configuration change due to such a damage) and for further developments of current NNs.

Conclusions

This document concerns the development and practical demonstration of a powerful innovative approach for aerodynamic angle estimation on flight vehicles using virtual air data sensors based on the neural predictive techniques. Two neural networks were developed with the aim of predicting the angles of attack and sideslip, by processing inertial data, command surface positions, flap positions and dynamic pressure. The accuracy target of ± 1 *deg* maximum error (with reference to free stream conditions) was considered as the fail-pass criterion. The developed NN-based virtual sensors demonstrated to be accurate over the entire flight envelope of two mathematical aircraft models. The De Havilland DHC-2 "Beaver" implemented in the Matlab FDC toolbox and the Alenia Aermacchi Sky-Y simulator were used during the course of this work to train and test the neural networks.

One of the purpose of this work was to establish a general training methodology that could be used for next real applications. During the training stage, several maneuvers were in fact simulated by reproducing those performed by pilots during the real Sky-Y flight tests. Conversely, test maneuvers were defined to simulate both flight conditions inside the aircraft flight envelope and extreme flight conditions not achievable during airplane operative life and not expected to be in the training data set. The aim of the test stage was therefore to stress the NNs as much as possible in order of evaluating the neural network performance both in realistic and extreme flight conditions and to evaluate any possible diverging behaviour of NN predictions. The best training and test strategies were set up using the Matlab FDC toolbox; moreover, the same simulator was also used to define the NN architecture for applications on the Sky-Y simulator.

The test of virtual sensors using the Alenia Aermacchi Sky-Y simulator, showed that

the neural networks estimated aerodynamic angles with acceptable errors (lower than ± 1.0 *deg*) when flight simulations were performed within the training boundaries with light and medium air turbulence according to the Dryden turbulence model. For the purpose of this work, using a flight simulator, rather than a real aircraft, allowed to test NNs in some flight conditions not obtainable in real operations or outside the aircraft flight envelope (e.g. high dynamic maneuvers) as done for test maneuvers in this work. Beyond the training boundaries and outside the real flight envelope, when high dynamic maneuvers were performed at speeds of up to V_D, the NNs continued to work properly with maximum absolute errors of less than 1 *deg*. In order to reproduce more realistic flight conditions, NNs were also tested simulating, in a conservative way, realistic electronic noise of sensors which provide the neural networks with the input signals. Even under these circumstances, the NN predictions resulted to be acceptable. The presence of noisy signals in the NN input vectors in fact produced very small additional errors of less than 0.1 *deg*.

At the Sky-Y integration rig of Alenia Aermacchi there was the chance to test virtual sensors using the Sky-Y simulator with a real FCC within the simulation loop. Virtual sensors, programmed in Matlab, were translated into $C++$ language and then adequately compiled and downloaded on real FCC. This activity allowed to define a process from concept (essentially a Matlab-assisted design) to realization of a final software usable on the real Sky-Y flight computer, and to investigate virtual sensor performance when the NNs were running on the real FCC. This activity highlighted that additional errors were introduced in the NN predictions because the current FCC does not work in double precision like Matlab does. The maximum errors were therefore beyond the tolerance limits in some circumstances. The virtual sensors estimated aerodynamic angles with maximum absolute errors of up to 1.2 *deg* for α and 1.5 *deg* for β, when high dynamic maneuvers were performed for speeds of up to V_D or for medium turbulence. Although these latter results are outside the prescribed tolerance limits, the following two considerations should be made. Firstly, tests were carried out beyond the training boundaries in order to evaluate the neural network performance outside the real aircraft flight envelope and mainly to investigate the occurrence of diverging outputs. Therefore, maximum errors of

less than ± 1.5 *deg* confirmed that NNs do not diverge and the NN predictions can be still considered reliable aerodynamic angles estimation even during extreme flight conditions. Secondly, within the flight envelope, the points lying outside the acceptance limits are mainly due to highest dynamic maneuvers, which were achieved using step commands (never used during training stages), during medium turbulence. It was noted that, in the same conditions, piloting the aircraft using a conventional pilot's control stick, the maximum errors were again within the tolerance limits. It is obvious that more accurate NNs could be obtained both for α and β by incorporating all the possible flight conditions, even extreme ones, inside the training set. However, in this work, a realistic training strategy was adopted to define a general methodology that could be used as a training technique for real aircraft. Overall, NN predictions were outside tolerance limits during particular flight conditions (e.g. $V > V_D$ and/or using step commands) that were not actually achievable during normal flight of the UAV aircraft considered here. The NNs, which were developed initially for Beaver application and then for Sky-Y simulator, may therefore be considered for tests on real Sky-Y unmanned airplane.

In this work, neural network models were adapted to the particular aircraft considered here, while the techniques used during the development and application of these neural models are generally applicable to any kind of real aircraft, manned or unmanned. Therefore, the training techniques, the signal processing systems developed and results obtained during this study may also act as a guide for the fundamental aspects of the further developments of current and future virtual sensors based on soft computing techniques.

Overall, the test stage of the virtual sensors using real FCC highlighted:

- NNAAEs were able to accurately generalize within the real aircraft flight envelope exploiting several maneuvers inspired to those performed by pilots to collect flight data for air data system calibration. NNs in particular worked properly during high dynamic maneuvers, and for light and medium turbulence, if no step commands are used, as occurs on real aircraft.

- NNAAEs can still be considered suitable estimators of α and β outside the training boundaries (aircraft speed up to V_D) if a greater error is accepted. In

fact, NNs were proven to be able to extrapolate outside the training boundaries, without diverging, and with maximum errors limited to ± 1.5 *deg*.

- NNAAEs were tolerant to realistic electric noise on the input signals, which, in the present case, was simulated according to the available literature on current inertial sensors and dynamic pressure transducers.
- The architecture of NNAAEs was not very sensitive to specific aircraft and those used in this work can easily be re-used as starting points for other aircraft with similar flight envelopes to "Beaver" and Sky-Y aircraft.

A general training strategy was identified during the training stage in terms of dedicated maneuvers adequate to be flown by real aircraft, even during the flight tests for the air data calibration, and able to create a training data set sufficiently representative of the aircraft dynamics. Both BP and LM training algorithms were proven to be suitable for training neural networks for the estimation of aerodynamic angles and for use in modern workstations; however, the LM algorithm showed a higher speed of convergence and was therefore used throughout the present work. Moreover, thanks to the several re-training sessions, by using random initialization of neural synaptic weights, local minima did not reveal real obstacles for this activity. In the last chapter of this work, a sensitivity analysis was presented in which the accuracy measurements of NN inputs was considered and several failure modes of input signals were simulated. The results seemed to suggest that both inputs for NNA and NNB could be reduced without affecting the neural network performance to any extent. However, the new proposed NN input vectors were not further investigated, because the full input vectors in NN software allowed other future investigations of the present virtual sensors.

In fact, even though the virtual sensors were successfully downloaded and tested on real hardware, there are some issues that need to be addressed before NN-based virtual sensor could be operatively used on a real aircraft.

During the present activity, one of the most important issue emerged when noisy signals were processed by artificial neural networks. If the noise grew over certain limits, NNs produced non acceptable estimations of the aerodynamic angles. Such

noise levels could be achieved when considering the effect of structural vibrations on inertial sensors of a particular aircraft, for example. In this work, high noise levels were only simulated to evaluate their effects on errors produced on neural network estimations. In fact, the analysis of noise deriving from structural vibrations depends on the particular case at hand and the structural vibration noise is usually reduced using classic methods, such as notch filters. Other unconventional techniques may be considered (e.g. NF) in order to evaluate the ability to filter noise without introducing a time delay as common filters do.

As far as the aeronautical certification needed by NNAAE to be safely used on aircraft is concerned, no real obstacles were identified. Since a trained neural network is a deterministic data processor, e.g. a software, NNs could be certifiable following current procedures for aeronautical software. Moreover, the standard procedure for selection of the NN architecture and the training strategy, which was defined through this work, retraces other common calibration procedures already certified in the aeronautical field.

Finally, although some issues emerged throughout this work, they do not seem to represent blocking points for the future applications on UAV and further developments. In fact, since the virtual sensors based on NNs were already tested and validated on real hardware (FCC), the software can now be considered ready to be implemented in real time on a test aircraft in order to uncover any characteristics of the virtual sensors that have not been discovered yet.

This page intentionally left blank.

Thanks

A success cannot be pursued without much help and a pinch of luck.
As a phD student, I had to move to Torino. Before leaving home I was so scared, but I met new friends who made me feel at home and many people who helped my PhD research to be a success.
First of all, I would like to thank Piero and Manuela with all my heart. My thanking them for their continuous and priceless support to my research may seem obvious, but they were the first people who welcomed me in Torino, and they never made me feel too far from home. I still remember when I moved and they invited me to have a pizza with them, and just few days later to have dinner with their own family: surprisingly I already had a new family!
Thanks so much to Silvio for his important role during these three years, not only for his knowledge and experience about ADS, but also for his huge passion about aviation in general. His spirit always pushed me to do more and better.
At Politecnico di Torino, I met other PhD students, some of them soon became new friends. Particular thanks to Matteo for his omniscience which always helped me during these three years.
In Alenia Aermacchi I met lots of people that helped my research, without their work I would not be here to write these lines. I would like to thank Mr. Paolo Gianardi and Mr. Massimo Maroni, who succeeded as head of FCS team, because both of them followed with extraordinary interest my research and supported my academic growth. In particular, Mr. Gianardi let me to spend a period abroad in the US as visiting scholar. Thanks to that experience I was able to go deeper into my academic studies and it also was very important for my own growth.
Moreover, I would like to thank Mr. Magaosso, Mr. Pusceddu, Mr. Caruso, Mr.

Bonardo, Mr. Chiavola, Mr. Binello of Alenia Aermacchi for their inestimable help in coding, testing and debugging NN codes I developed.

During the six months I spent overseas, I met wonderful people that I will never forget for their help and support. First of all, my "nonni" Russ and Gail. I was welcomed in their home as a grand-son. Living with them was so important to appreciate the US-lifestyle otherwise I would have never known. The happy hours spent together will always live in my mind. How could I forget Sundays spent together or dominoes battles with Donna? A special thank to the whole Ireland family, for making me feel as part of them. In particular, Jonh Ireland whose interest in my research sustained myself during a "flop" period while programming NN software.

At UCLA I had the opportunity to live what I had always seen as a dream. In particular I would like to sincerely thank prof. Ivan Catton because without his support I would have never been able to go to UCLA. In his lab I met several guys, Krsto and David became friends out of the lab, thank you so much for your support and friendship. I will never forget my roommate Myles, my big bro', his advice still echoing in my mind, his songs in my life. Thank you for everything you did for me and all time spent together, but believe me: I will never build a cylon!

Three years in a new town without my new friends would have been impossible. Real friendship was born with Gianluca (Scoglio) and Cristina (Griss), a friendship I will always preserve as a treasure: they soon became my new family where taking shelter. Thanks to Scoglio for all the nights spent playing "granatu" and building aircraft models and to Griss for cooking and putting up with us for the mess we made every time! But that family soon enlarged: Giuseppe (Peppino), Carlo (Carlé), Carolina (Carol) and baby Eva. Their friendship is so important that I cannot imagine my life in Torino without them!

A special thank to Alessandro and Davide, our time spent together is really unforgettable, isn't it?.

For sure living far from home made me better estimate ties with my family and old friends.

I would like to thank people who have always supported me: my family and my

friends from Caserta. I have no words to thank my parents, my sister and aunt Maria for everything they have done during these three long years. Thanks for doing up my house like new and my sister for helping me several times while I was living “very messy times”!

I felt my old friends even closer when I moved to Torino, thank you so much to Michele (Usá) for supporting myself when running down in particular.

I met lots of new people, but some others were lost and forced me to become stronger and stronger to overcome their absence: they will all continue to live in my heart and my prayers.

I hope to have been able not forget all friends and people who remarkably marked my life in the last three years, making my research possible and my life a better reality.

Bibliography

[1] W.L. Ikard and United States. National Advisory Committee for Aeronautics. *An Air-flow-direction Pickup Suitable for Telemetering Use on Pilotless Aircraft.* Technical note // National Advisory Committee for Aeronautics. National Advisory Committee for Aeronautics, 1956.

[2] S.H. Chue. Pressure probes for fluid measurement. *Progress in Aerospace Sciences*, 16(2):147 – 223, 1975.

[3] R.C. Pankhurst and D.W. Holder. *Wind-tunnel technique: an account of experimental methods in low- and high-speed wind tunnels.* Pitman, 1952.

[4] K. S. Yajnik and R. P. Gupta. A new probe for measurement of velocity and flow direction in separated flows. *Journal of Physics E Scientific Instruments*, 6:82–86, January 1973.

[5] A. Calia, R. Galatolo, V. Poggi, and F. Schettini. Multi-hole probe and elaboration algorithms for the reconstruction of the air data parameters. In *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pages 944 –948, 30 2008-july 2 2008.

[6] J.E. Tomayko, United States. National Aeronautics, and Space Administration. History Office. *Computers take flight: a history of NASA's pioneering digital fly-by-wire project.* The NASA history series. NASA, 2000.

[7] Marcello R. Napolitano, Younghwan An, and Brad A. Seanor. A fault tolerant flight control system for sensor and actuator failures using neural networks. *Aircraft Design*, 3(2):103 – 128, 2000.

[8] M. Oosterom and R. Babuska. Virtual sensor for fault detection and isolation in flight control systems - fuzzy modeling approach. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2645 –2650 vol.3, 2000.

[9] M. Oosterom and R. Babuska. Virtual sensor for the angle-of-attack signal in small commercial aircraft. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 1396 –1403, 0-0 2006.

[10] P. Samara, G. Fouskitakis, and S. Sakellariou, J.and Fassois. Aircraft angle-of-attack virtual sensor design via a functional pooling narx methodology. In *European Control Conference, University of Cambridge, UK, 2003.*, 2003.

[11] Xiaoping Du, Huamei Sun, Kun Qian, Yun Li, and Liantao Lu. A prediction model for vehicle sideslip angle based on neural network. In *Information and Financial Engineering (ICIFE), 2010 2nd IEEE International Conference on*, pages 451 –455, sept. 2010.

[12] T. J. Rohloff, I. Catton, and S. A. Whitmore. Simulation of a Flush Air-Data System for Transatmospheric Vehicles. *Journal of Spacecraft and Rockets*, 45:716–732, July 2008.

[13] Ihab Samy, Ian Postlethwaite, Da-Wei Gu, and John Green. Neural-Network-Based Flush Air Data Sensing System Demonstrated on a Mini Air Vehicle. *Journal of Aircraft*, 47:18–31, 2010.

[14] Kevin A. Wise. Computational air data system for angle-of-attack and angle-of-sideslip, 08 2005.

[15] K. McCool and D. Haas. Neural network system for estimation of aircraft flight data, 10 2002.

[16] J. Svobodova, V. Koudelka, and Z. Raida. Aircraft equipment modeling using neural networks. In *Electromagnetics in Advanced Applications (ICEAA), 2011 International Conference on*, pages 627 –630, sept. 2011.

[17] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions On Pattern Analysis and Machine intelligence*, 20:23–38, 1998.

[18] Ministero della Difesa Direzione Generale degli Armamenti Aeronautici. *Omologazione di Aeromobili Militari e Relativi Sistemi.* 2006.

[19] Ministero della Difesa Direzione Generale degli Armamenti Aeronautici. *Istruzioni per la Compilazione dei Capitolati Tecnici per Aeroplani Prototipi e di Serie.* 2005.

[20] W. Gracey. *Measurement of aircraft speed and altitude.* NASA reference publication. Wiley, 1981.

[21] W. Wuest, North Atlantic Treaty Organization. Advisory Group for Aerospace Research, and Development. Flight Mechanics Panel. *Pressure and Flow Measurement.* AGARDograph: AGARD flight test instrumentation series. North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development, 1980.

[22] Edward A. Haering Jr. Airdata measurement and calibration. *NASA/TM-1995-104316*, 1995.

[23] Mark C. Davis, Joseph W. Pahle, John Terry White, Laurie A. Marshall, Michael J. Mashburn, Rick Franks, and Sverdrup Corp. Development of a flush airdata sensing system on a sharp-nosed vehicle for flight at mach 3 to 8. *NASA/TM-2000-209017*, 2007.

[24] A.J. Meade A. Srivastava and K. Long. Learning airdata parameters for flush air data sensing systems. *IEEE Transactions On Systems, Man, And Cybernetics*, 2010.

[25] T.J. Larson and Ames Research Center. *Qualitative evaluation of a flush air data system at transonic speeds and high angles of attack.* NASA technical paper. National Aeronautics and Space Administration, Scientific and Technical Information Branch, 1987.

[26] Stephen A. Whitmore, Brent R. Cobleigh, and Edward A. Haering. Design and calibration of the x-33 flush airdata sensing (fads) system. 1998.

[27] Stanford University. Joint Institute for Aeronautics, Acoustics, C.S. Lee, and N.J. Wood. *Calibration and data reduction for a five-hole probe.* 1986.

[28] Air Force 11. Pitot and static pressure systems. installation and inspection of. *Military Specifications, MIL-P-26292C (USAF)*, 30 Sept. 1988.

[29] J. McCarthy, M.L. Minsky, C.E. Shannon, N. Rochester, and Dartmouth College. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence.* 1955.

[30] E. Rich and K. Knight. *Artificial intelligence.* Artificial Intelligence Series. McGraw-Hill, 1991.

[31] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: symbols and search. *Commun. ACM*, 19(3):113–126, March 1976.

[32] Lotfi A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Commun. ACM*, 37(3):77–84, March 1994.

[33] S.R. Cajal, P. Pasik, T. Pasik, P. Pasik, and T. Pasik. *Texture of the Nervous System of Man and the Vertebrates.* Number v. 3 in Texture of the Nervous System of Man and the Vertebrates: An Annotated and Edited Translation of the Original Spanish Text with the Additions of the French Version. Springer, 2002.

[34] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[35] WarrenS. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[36] Robert Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of IEEE First Annual International Conference on Neural Networks*, volume 3, pages III–11–III–14, 1987.

[37] R. Hecht-Nielsen. *Neurocomputing.* New Horizons in Technology Series. Addison-Wesley Publishing Company, 1990.

[38] R. Lippmann. An introduction to computing with neural nets. *ASSP Magazine, IEEE*, 4(2):4 –22, apr 1987.

[39] David A. Sprecher. A universal mapping for kolmogorov's superposition theorem. *Neural Netw.*, 6(8):1089–1094, January 1993.

[40] Federico Girosi and Tomaso Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Comput.*, 1(4):465–469, December 1989.

[41] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[42] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.

[43] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Netw.*, 2(3):183–192, May 1989.

[44] M.A. Arbib. *Brains, machines, and mathematics.* Springer-Verlag, 1987.

[45] N. Kumar. *Comprehensive Physics for Engineers.* Laxmi Publications Pvt Limited, 2005.

[46] J.A. Hertz, A.S. Krogh, and R.G. Palmer. *Introduction To The Theory Of Neural Computation.* Santa Fe Institute Studies in the Sciences of Complexity: Lecture Notes. Westview Press, 1991.

[47] M.T. Hagan, H.B. Demuth, and M.H. Beale. *Neural Network Design.* Electrical Engineering Series. Pws Pub., 1996.

[48] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 1999.

[49] J.S.R. Jang, C.T. Sun, and E. Mizutani. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence.* MATLAB curriculum series. Prentice Hall, 1997.

[50] M. Nørgaard. *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook.* Advanced Textbooks in Control and Signal Processing. Springer, 2000.

[51] J. M. Mendel and R. W. McLaren. A prelude to neural networks. chapter Reinforcement-learning control and pattern recognition systems, pages 287–318. Prentice Hall Press, Upper Saddle River, NJ, USA, 1994.

[52] A. E. Bryson and YâĂŞC. Ho. Applied optimal control: optimization, estimation, and control. *Taylor & Francis*, 1969.

[53] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, 1974.

[54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[55] Robert A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295 – 307, 1988.

[56] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Applied Math.*, 2:164–168, 1944.

[57] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[58] Stephen M. Goldfeld, Richard E. Quandt, and Hale F. Trotter. *Maximization by Quadratic Hill-Climbing*, volume 34. The Econometric Society, 1966.

[59] Stefano Ghirlanda and Magnus Enquist. A century of generalization. *Animal Behaviour*, 66(1):15 – 36, 2003.

[60] M.A. Arbib. *Handbook of Brain Theory and Neural 2e.* Bradford Bks. Mit Press, 2003.

[61] CD-Adapco. *Star-CCM+, User Guide.*

[62] Sensing Honeywell and Control. Integrated pressure transducer. ADS-14184, 2012.

[63] M. O. Rauw. Fdc 1.2 a simulink toolbox for flight dynamics and control analysis. 2001.

[64] B.L. Stevens and F.L. Lewis. *Aircraft Control and Simulation.* John Wiley & Sons, 2003.

[65] B. Etkin and L.D. Reid. *Dynamics of Flight: Stability and Control.* Wiley, 1995.

[66] Military specification mil-f-8785c.

[67] Bosch Sensortec. Bma150 digital, triaxial accleration sensor. 2010.

[68] Bosch Sensortec. Bmp180 digital pressure sensor. 2012.

[69] STMicroelectronics. L3gd20 mems motion sensor: three-axis digital output gyroscope. 2011.

[70] muRata. The sca100t dual axis inclinometer series.

[71] F. Stoliker. *Introduction to Flight Test Engineering.* AGARDograph: AGARD flight test instrumentation series. AGARD-AG-300. North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development, 1995.

[72] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models.* Wiley, 2004.

[73] D.G. Cacuci. *Sensitivity & Uncertainty Analysis, Volume 1: Theory.* Sensitivity and uncertainty analysis. Taylor & Francis, 2003.

[74] L.M. Reyneri. Tutorial: About industrial acceptance of intelligent systems. In *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pages 964 –969, 30 2009-dec. 2 2009.

[75] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: The Primer*. Wiley, 2008.

[76] Military handbook mil-hdbk-1797.